The exframe Package

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

nbeisert@itp.phys.ethz.ch

2024/10/18, v3.5

Abstract

exframe is a LaTeX 2ε package which provides a general purpose framework to describe and typeset exercises and exam questions along with their solutions. The package features mechanisms to hide or postpone solutions, to assign and handle points, to collect problems on exercise sheets, to store and use metadata and to implement a consistent numbering. It also provides a very flexible interface for configuring and customising the formatting, layout and representation of the exercise content.

Contents

1 Introduction

This package provides a framework to describe and typeset exercises (homework problems, classroom exercises, quizzes, exam questions, exercise questions in books and lecture notes, ...) and their solutions or answers. The aim of this package is to set up a few LaTeX environments into which questions and corresponding answers can be filled conveniently. The main task of the package is to manage the text and data that are provided in the source document, perform some common operations on them, and then output the content appropriately. The package has the following goals, tasks and features:

- The package is designed with generality in mind. It is meant to be usable in many different situations. The primary target is science and education, but it may well be useful in other areas.
- The package defines a basic functional layout for the output and provides many options to reshape the layout and formatting according to the author's needs and wishes.
- The package can handle two layers of exercises: main problems and subproblems. The use of subproblems is optional.
- The display of solutions can be configured: Solutions can be hidden for a hand-out version of exercise sheets. When displayed, they may appear immediately, collectively after the problem, at the end of each sheet or at some manually defined location.
- The package can handle exercise sheets which combine several exercise problems: A LaTeX document can consist of an individual sheet or of a collection of sheets (e.g. spanning a lecture course). In the latter case, the document files can be set up such that single sheets as well as a collection of all sheets can be compiled; the package childdoc may be of assistance.
- The package can handle points to be credited: Points will be displayed according to the layout. Overall points for a problem or a sheet can be added automatically. Points

can also be stored and used elsewhere.

- The package provides an interface to specify exercise metadata (author, source, ...): Some basic types of metadata are predefined and more specific metadata categories can be added.
- The package can use alternative counters for equations within solutions (and problems). This is to ensure a consistent numbering independently of whether solutions are output or not.

2 Usage

To use the package exframe add the command

```
\usepackage{exframe}
```

to the preamble of the LaTeX document.

2.1 Exercise Environments

The package provides four environments to describe the main entities of exercise problems. Additional information on the exercises can be provided in the optional arguments to these environments which will be discussed in the following sections. Furthermore, a limited set of commands is provided for control and extra features, see the sections below for details.

`problem` (*env.*) The `problem` environment describes an exercise problem:

```
\begin{problem}[opts]
  problem text and subproblems
\end{problem}
```

As one of the many available options *opts*, one can provide a title for the exercise by specifying `title={`*title*`}`. If no title is given, the problem number will be displayed instead. See section 2.3 and section 2.4 for a description of the available options.

`subproblem` (*env.*) The `subproblem` environment describes a subproblem, part or an individual question of an exercise problem:

```
\begin{subproblem}[opts]
  subproblem text
\end{subproblem}
```

A `subproblem` environment must be contained within a `problem` environment (however, a `problem` block need not contain `subproblem` blocks).

`solution` (*env.*) The `solution` environment describes the solution to a problem or a subproblem:

```
\begin{solution}[opts]
  solution text
\end{solution}
```

A `solution` environment should be at the end of a `subproblem` or `problem` environment (it is not mandatory to provide a `solution`). It can be contained within the corresponding environment or it can follow it (where needed, the option `forproblem` associates a `solution` following a `subproblem` to the encompassing `problem` rather than the preceding

subproblem). Depending on the choice of solution display, see section 2.2, the output may have a slightly different layout. In terms of logic, it is preferred to define a solution *within* the corresponding environment; this may also have some technical advantages and produce a slightly better result in terms of layout.

`sheet` (*env.*) The `sheet` environment describes an exercise sheet:

```
\begin{sheet}[opts]
  sheet text and problems
\end{sheet}
```

A sheet typically contains one or several problems (it is not mandatory to group problems into a `sheet`). There may or may not be additional auxiliary text introducing the problems. A header will be added to the sheet according to the specified layout.

2.2 Solution and Problem Display

There are several options to control the output of solutions and of problems.

`solutions` Most importantly, the display of solutions can be disabled or enabled altogether:

```
\exercisesetup{solutions[=true|false]}
```

Solutions are hidden by default, and their display needs to be activated explicitly (it suffices to specify the option `solutions` without the value `true`). It is also possible to control the display by an analogous package option `solutions`, see section 2.8 for further information.

`\ifsolutions` `onlysolutions` The display of solutions is reflected by the conditional `\ifsolutions`. As the hiding of solutions is performed automatically, the conditional would typically be used to change some details, e.g. for adjusting titles:

```
\ifsolutions Solutions\else Exercises\fi
```

Alternatively, content to be processed only in solutions mode can be enclosed in an `onlysolutions` block:

```
\begin{onlysolutions}
...
\end{onlysolutions}
```

This structure can be useful to hide auxiliary text or material if all solution content is to be stripped from a source file, e.g. by an automated sed filter rule (doubling of backslashes required for sed as well as for shell script strings):

```
sed "/\\\\begin{solution}/,/\\\\end{solution}/d;"\
"/\\\\begin{onlysolutions}/,/\\\\end{onlysolutions}/d"
```

`solutionequation` As solutions can contain numbered equations while the display of solutions can be switched on and off, it is important to assign a different counter for equations within solutions in order for the equation numbers to be stable. A separate counter for equations within solutions is enabled by default. It can be disabled by:

```
\exercisestyle{solutionequation=false}
```

This option prepends the letter 'S' to equation numbers within solutions which are counted separately; the display can be configured differently, see section 2.6.

`solutionbelow` `\insertsolutions` The package allows to collect solutions and defer their display to particular locations:

```
\exercisestyle{solutionbelow=pos}
```

The available choices for *pos* are to display solutions where they are defined (`here`), defer them to the end of the current subproblem (`subproblem`), problem (`problem`) or sheet (`sheet`) or display them at a manually chosen location (`manual`). Note that typically solutions are defined at the end of a (sub)problem and therefore the choice `here` is similar to (`sub`)`problem`. The latter form, however, makes sure that a solution does not inherit the margin of the parent environment. The alternate modes `problem*` and `subproblem*` positions the solution *after* the (sub)problem environment such that it does not inherit any layout, but also no definitions made in the parent environment. In `manual` mode, all solutions are collected (with appropriate headers) until they are output by the directive `\insertsolutions`. If no solutions are stored in the buffer (or if the mode is not `manual`), `\insertsolutions` has no effect.

`\writesolutions` Another option to handle solutions is to write them to a file for later use. Writing to a file is initiated by:

```
\writesolutions[filename]
```

The optional argument describes the filename as *filename*`.sol`; no argument defaults to the main tex filename as `\jobname.sol`; the extension `.sol` can be customised by the configuration `extsolutions`. This mode overrides the `solutionbelow` behaviour described above; all subsequent solutions are written to the file. The file is closed by `\closesolutions` and the display of solutions returns to manual mode. It is not necessary to close a file as it will be closed automatically by reading from a file, writing to another file or by the end of the document.

`\readsolutions` Solutions are read from a file by:

```
\readsolutions[filename]
```

This command outputs a sectional title and reads the file via `\input{`*filename*`.sol}`.

`solutionbuf` `problembuf` `subproblembuf` The package offers similar functionality to control the display of problems. In order to have any control over the content of `problem` and `subproblem` environments, the latter need to be read into an internal buffer. Reading of solutions and problems to internal buffers is activated or deactivated by:

```
\exercisesetup{solutionbuf[=true|false]}
\exercisesetup{problembuf[=true|false]}
\exercisesetup{subproblembuf[=true|false]}
```

By default, `solution` environments are read to an internal buffer, while the content of [`sub`]`problem` environments is processed directly by the TEX engine. Therefore, the below options to control the display of [`sub`]`problem` environments require the statement `\exercisesetup{[sub]problembuf}`.

`problemmanual` `\insertproblems` The immediate display of `problem` environments is controlled by:

```
\exercisestyle{problemmanual[=true|false]}
```

In the default automatic mode, problems are displayed directly where they are declared. In manual mode, problems are collected to an internal buffer, and only displayed by issuing `\insertproblems`.

Note that `solution` environments should be declared within the corresponding `problem` environment in order to preserve their appropriate association. The `solution` environment

is then processed at the place where the `problem` environment is displayed, and it may (or may not) be deferred further.

`\writeproblems` `\readproblems` Problems can be written out to an external file for later usage. The functionality is analogous to solutions and uses the macros:

`\writeproblems[`*filename*`]`
`\readproblems[`*filename*`]`

The optional argument describes the filename as *filename*`.prb`; no argument defaults to the main tex filename as `\jobname.prb`; the extension `.prb` can be customised by the configuration `extproblems`.

`disable` `insertproblemselect` `sertsubproblemselect` The display of a particular [`sub`]`problem` can be suppressed altogether by an optional argument:

`\begin{[sub]problem}[disable]`

This option can be exploited to automatically suppress certain classes of problems as follows: A hook function `insert`[`sub`]`problemselect` declared by:

`\exerciseconfig{insert[sub]problemselect}[1]{`*code*`}`

can call `\set`[`sub`]`problemdata{disable}` whenever a problem is to be suppressed. In order to decide, the optional argument of the [`sub`]`problem` environment is passed on to the hook function as the single argument. Note that the argument needs to be processed manually.

2.3 Metadata

In a collection of exercise problems it makes sense to keep track of metadata for the overall collection as well as for individual problems and potentially display some of them. The framework defines a standard set of metadata fields and offers functionality to add more specialised metadata fields.

`\exercisedata` Global metadata is specified by the command:

`\exercisedata{`*data*`}`

The argument *data* is a comma-separated list of metadata specifications in the form *key*`={`*value*`}`. The standard set of global metadata keys consists of:

- `author`: principal author(s) of the exercise collection; also invokes the LaTeX command `\author`; will be written to pdf documents.
- `title`: title of the exercise collection; also invokes the LaTeX command `\title`; will be written to pdf documents.
- `date`: date of the exercise collection; also invokes the LaTeX command `\date`; will be written to pdf documents.
- `subject`: subject area of the exercise collection; will be written to pdf documents.
- `keyword`: keyword(s) for the exercise collection; will be written to pdf documents.
- `course`: title of the course (class, lecture, module, ...) for the exercise collection.
- `institution`: institution (school, department, institute, university, ...) offering the course or exercise collection.

- `instructor`: instructor(s) for the course or exercise; this field refers to person(s) who organise the corresponding course or exercises whereas `author` refers to the principal creator of the material.
- `period`: period (year, season, date, term identifier, ...) of the corresponding course.
- `material`: type of material (exercises, homework assignments, exam, quizzes, solutions, ...).

`\defexercisedata` Additional custom fields for global metadata can be created with:

`\defexercisedata{`*key*`}`

`\getexercisedata` `\exercisedataempty` Global metadata should typically be specified somewhere at the top of the main document, and it can be inserted wherever needed. There are two commands to read and process metadata. To insert the value of metadata field *key* use:

`\getexercisedata{`*key*`}`

In some situations the output should depend on whether a metadata has been filled (e.g. to fill a default value or to display something else instead). This can be checked with the conditional:

`\exercisedataempty{`*key*`}{`*empty code*`}{`*filled code*`}`

The *empty code* is executed if no value or an empty value has been specified; otherwise the *filled code* is executed.

`sheet` `problem` The package offers a similar mechanism to describe and use metadata for sheets and problems:

`\begin{sheet}[`*opts*`]`
`\begin{problem}[`*opts*`]`

The argument *opt* is a comma-separated list which can contain metadata specifications in the form *key*`={`*value*`}`. The standard set of metadata keys for sheets consists of:

- `due`: indication of the due date for the exercise sheet.
- `handout`: indication of the handout date for the exercise sheet.
- `title`: specifies a title for the sheet; when reading value (see below), returns composed title; untitled sheets will be displayed by their number; title will be written to pdf documents.
- `rawtitle` (for reading only): contains the raw title as specified by `title`.
- `author`: author(s) of the sheet; will be written to pdf documents.
- `editor`: editor(s) of the sheet; this field refers to a person who makes adjustments to the sheet whereas `author` refers to the creator of the sheet.
- `editdate`: indication of the date when the sheet was last edited.

The standard set of metadata keys for problems consists of:

- `title`: specifies a title for the problem; when reading value (see below), returns composed title; untitled problems will be displayed by their number.
- `rawtitle` (for reading only): contains the raw title as specified by `title`.

\defsheetdata
\setsheetdata
\getsheetdata
\sheetdataempty
\defproblemdata
\setproblemdata
\getproblemdata
\problemdataempty

Metadata for sheets can be used in the same way as the global metadata. The following directives are analogous to `\defexercisedata`, `\exercisedata`, `\getexercisedata` and `\exercisedataempty`:

```
\defsheetdata{key}
\setsheetdata{data}
\getsheetdata{key}
\sheetdataempty{key}{empty code}{filled code}

\defproblemdata{key}
\setproblemdata{data}
\getproblemdata{key}
\problemdataempty{key}{empty code}{filled code}
```

pdfdata
\writeexercisedata

The most relevant metadata can be written to the metadata section of pdf files (using pdfLaTeX and the package hyperref whenever loaded). This feature is configured by:

```
\exercisesetup{pdfdata[=auto|manual|sheet|off]}
```

The option `auto` writes the global metadata `title`, `author`, `subject` and `keyword` to the corresponding fields in the pdf file. To make this work, these must be defined before the `\begin{document}` directive. The option `manual` allows to manually write these metadata by the command `\writeexercisedata`. It should be issued after the metadata have been set, but before any content is written to the pdf file. In other words, it can be anywhere in the document preamble directly after `\begin{document}`, or following a couple of content-free definitions at the beginning of the document body (in case the metadata should be set within the document body for some reason). The option `sheet` writes out the metadata at the beginning of the first `sheet` environment (which should follow `\begin{document}` without any content in between). This option is primarily for filling the `author` and `title` fields with metadata of a sheet rather than a collection of exercises. Note that if no `author` is defined for the sheet, the global metadata `author` is used. The option `off` disables all writing of metadata.

problem
subproblem
solution

There is an additional mechanism to keep track of metadata for problems, subproblems and solutions which can be displayed in the opening line of these entities. Displayed metadata serve two purposes: they are used to describe the quality of a problem or they are intended for internal documentation purposes. Their output can be controlled individually, e.g. only in development versions of a document. Note that specifying a key more than once will display the content multiple times in the order in which they are encountered. Displayed metadata are specified at the top of the corresponding environment:

```
\begin{problem}[opts]
\begin{subproblem}[opts]
\begin{solution}[opts]
```

The standard set of displayed metadata keys consists of:

- `author`: author(s) of the problem (or subproblem, solution).
- `editor`: editor(s) of the problem; this field refers to a person who has made adjustments to the problem whereas `author` refers to the creator of the problem.
- `source`: source of the problem; in case the problem has been taken from elsewhere (conceptually or literally).
- `difficulty`: indication of the level of difficulty of the problem.
- `keyword`: keyword(s) for the problem;

- `comment`: some comment on the problem.
- `optional` (display enabled by default): whether addressing the problem is mandatory or optional; by default the text will be displayed after the title in italic shape.

By default, only the `optional` items are displayed, all other types of items are hidden; controlling the display for each type of item is described below.

`extdata` Further displayed metadata keys are defined by the package option `extdata`, see section 2.8:

- `review`: field to review the aspects of the problem (quality, length, appropriateness, difficulty, ...).
- `recycle`: indication of previous instances where this problem was used.
- `timesolve`: indication of the time needed to solve this problem (or subproblem).
- `timepresent`: indication of the time needed to present this problem (or subproblem, solution).

`\showprobleminfo` The display of the above metadata fields for a problem (or subproblem, solution) is controlled by:

`\showprobleminfo{`*keys*`}`

Here *keys* is a comma-separated list of keys to be activated (*key* or *key*`=true`) or deactivated (*key*`=false`).

`\defprobleminfo` Displayable metadata can be defined or adjusted by:

`\defprobleminfo{`*key*`}{`*code*`}`

Here *key* specifies the metadata field and *code* the code to display this type of metadata where the argument `#1` represents the data to be displayed.

`insertprobleminfo` `insertsubprobleminfo` `insertsolutioninfo` `\addprobleminfo` `\addprobleminfo*` Additional information can be injected into the opening line of problems and solutions by the definitions:

```
\exerciseconfig{insertprobleminfo}{code}
\exerciseconfig{insertsubprobleminfo}{code}
\exerciseconfig{insertsolutioninfo}{code}
```

The hook code *code* will be called after processing the environment arguments. Information can be added to the opening line by:

```
\addprobleminfo{info}
\addprobleminfo*{info}
```

The unstarred command adds information at the end of the opening line, the starred version at the beginning (but after the title or identifier).

2.4 Points

`points` Exercise problems or certain parts of them can be credited with points (credits, awards, ...). The package provides an interface to specify and manage such points. Points are declared by the option `points=`*points* for the environments `sheet`, `problem` and `subproblem`. These numbers will be printed to the opening line of problems and subproblems.

Note that the points should normally be integer numbers. Fractional points are permissible as well, but the internal storage by the TEX engine is somewhat limited, so that only fractions

with powers of two as denominators (.5, multiples of .25, .125, .0625, ...) are reliable. More general fractional decimal numbers such as multiples of 0.2 will be subject to rounding errors and will not display nicely.

Bonus points can be specified in the format `points=`[*regular*][`+`*bonus*]. By default, such points will be printed as [*regular*][`+`*bonus*] where 0 components are omitted.

`problempointsat` `subproblempointsat` `solutionpointsat`

The location where points of problems and subproblems shall be displayed can be adjusted individually by:

```
\exercisestyle{problempointsat=start|start*|margin|end|manual|off}
\exercisestyle{subproblempointsat=start|start*|margin|end|manual|off}
\exercisestyle{solutionpointsat=start|start*|margin|end|manual|off}
```

The default values are `start` and `end` for problems and subproblems, respectively. The option `start` displays points at the very end of the opening line; the option `start*` displays them at the start of it. The option `end` displays points at the end of the problem or subproblem text. The option `margin` displays points in the margin. The option `manual` displays points at a manually chosen location specified by the directive `\showpoints`. Note that `\showpoints` can also be used for the option `end` to display the points prematurely (e.g. if the text ends with a displayed equation, it may make sense to display the points just before the equation). The option `off` disables the display of points.

`\getsheetdata` Points for sheets are only stored by the package; they must displayed manually. Within the corresponding `sheet` environment the points can be accessed by:

```
\getsheetdata{points}
```

`\getsheetpoints` `\getproblempoints` `\getsubproblempoints` `\getsolutionpoints` `\extractpoints` `\switchpoints`

The package allows to read the point totals for other sheets and problems:

```
\getsheetpoints{[tag]}
\getproblempoints{[tag]}
\getsubproblempoints{[tag]}
\getsolutionpoints{}
```

Here *tag* is the tag assigned to the corresponding sheet or problem, see section 2.5. An empty argument *tag* refers to the current sheet, (sub)problem or solution. If bonus points are used, the points will be returned in the format [*regular*][`+`*bonus*]; the components *regular* and *bonus* can be extracted from the returned expression by `\extractpoints` and `\extractpoints*`, respectively. A convenient case switch of the returned value can be performed by:

```
\switchpoints{reg}{bonus}{both}{none}{val}
```

Here *val* is the value returned from the points register, *reg* is displayed for purely regular points, *bonus* is displayed for purely bonus points, *both* is displayed for mixed points, *none* is displayed for no points. In each of the four expressions, `#1` will be replaced by the regular points and `#2` by the bonus points.

`\awardpoints` Grading instructions with points to be awarded can be specified in the solution text by:

```
\awardpoints[details]{points}
\awardpoints*[details]{points}
```

Here *details* is an optional text with further details, e.g. to explain under which conditions these points are to be awarded. The starred form is used to specify optional points or alternative paths with alternative grading instructions. These points will be marked and not be used for the computation of a total.

`warntext` The package attempts to add up the points of subproblems to the problem total and likewise the points of problems to the sheet total. The package also performs some sanity checks on the provided numbers: If points are specified for both subproblems and problems or for both problems and sheets, they will be compared. Also the points within solutions (excluding optional or alternative points) are added up and compared to the corresponding problem or subproblem. Furthermore the package checks whether points are defined for all subproblems within a problem or all problems within a sheet. Mismatches are reported as package warnings. As point mismatches can be rather severe, there is an option to write such warnings directly into the output document (to be removed before distribution):

```
\exercisesetup{warntext[=true|false]}
```

`fracpoints` The package offers pretty display of fractional points with denominators 2, 4 and 8 by writing the decimal part as a fraction, e.g. $1.75 \rightarrow 1\frac{3}{4}$. This feature is enabled by:

```
\exercisestyle{fracpoints}
```

2.5 Labels and Tags

`label` LaTeX provides labels to make references to remote parts of the text. Labels can be set as usual by `\label{`*label*`}` within the `problem`, `subproblems` and `sheet` environments. Alternatively, they can be specified as the environment option:

```
label={label}
```

`tag` `\sheettag` `\problemtag` `\subproblemtag` The package provides an additional mechanism to tag sheets and problems. Each `sheet`, `problem` and `subproblem` can be assigned a unique tag *tag* by the environment option:

```
tag={tag}
```

This tag is used for reading point totals as described in section 2.4. Furthermore, the macro `\sheettag`, `\problemtag` or `\subproblemtag` is set to the tag *tag* within the current environment. If no tag is specified it defaults to the number of the current sheet or (sub)problem; note that this number can change by reordering sheets and problems and therefore it should not be used to identify the entity from other parts of the document.

A useful application for tags is to encapsulate labels within individual sheets and problems which are part of a collection of exercises. Labels which are composed as `\sheettag-`*label* or `\problemtag-`*label* can be considered local and will not clash with labels defined within a different environment. Within the same sheet or problem, local labels can be accessed by the same construction. They can also be accessed from remote parts of the document by fully expanding `\sheettag` or `\problemtag` for the desired target environment.

`autolabelsheet` `autolabelproblem` If unique tags are specified, the package can automatically create labels for sheets (`sheet:`*tag*) and problems (`prob:`*tag*) by:

```
\exercisesetup{autolabelsheet[=true|false]}
\exercisesetup{autolabelproblem[=true|false]}
```

`\getsheetlist` `\getproblemlist` `\getsubproblemlist` Tags can also be used to process a list of sheets, problems and subproblems:

```
\getsheetlist{}
\getproblemlist{[sheet-tag]|*}
\getsubproblemlist{[problem-tag]}
```

These commands return a list of sheets, problems and subproblems tags, where each item is encapsulated in braces. The commands `\get[sub]problemlist` return the (sub)problems within the specified sheet or problem. If no argument is given, the current sheet or problem is assumed. `\getproblemlist*` returns the list of all problems.

`\exerciseloop` `\exerciseloopstr` `\exerciseloopret` `exerciseloop`

The package defines two commands to walk through such a list:

```
\exerciseloop{items}{cmd}
\exerciseloopstr[ret]{items}{str}
```

Here, *items* is a list of items in braces, and `\exerciseloop` evaluates *cmd* for each item of the list where '`#1`' is replaced by the item. The command `\exerciseloopstr` concatenates the evaluations of *str* and returns the resulting string in the macro *ret* (which defaults to `\exerciseloopret`). Both commands maintain a counter of items `exerciseloop` which can be accessed within *cmd*/*str* or after `\exerciseloop[str]` to obtain the total number of items in the list.

2.6 Layout

The package provides a large number of parameters to adjust the display of exercises to a desired layout.

`\exerciseconfig` Configuration settings are declared and modified by the command:

```
\exerciseconfig{key}[narg]{value}
```

Here *key* is a key and *value* is its assigned value. Configuration options can also be macros with arguments in which case *narg* is the number of arguments and *value* is the macro definition using arguments `#n`. The command `\exerciseconfig` therefore is analogous to `\(re)newcommand` except that the definitions are encapsulated by the package and any previous definition is overwritten without checking.

`exerciseconfigappend` `xerciseconfigprepend` In some cases it may be useful to be able to append or prepend to a (parameterless) definition by:

```
\exerciseconfigappend{key}{value}
\exerciseconfigprepend{key}{value}
```

`\getexerciseconfig` `\exerciseconfigempty` Configuration definitions can be read by:

```
\getexerciseconfig{key}[arguments]
```

The number of arguments after `{`*key*`}` must match the optional argument *nargs* of the definition. Furthermore, it can be checked whether a configuration definition is empty:

```
\exerciseconfigempty{key}{empty code}{filled code}
```

The *empty code* is executed if no value or an empty value has been specified. Otherwise the *filled code* is executed.

The package defines numerous layout configuration options. They are listed along with their original definition and a brief description in section C.3. Some particular aspects are described in more detail in section 3.1. The configuration includes options to:

- adjust the language for the principal entities of this package like "sheet(s)", "problem(s)", "solution(s)", "points(s)";

- adjust the fonts styles of various parts of the text;
- adjust the spacing above, below, between various elements;
- define code to process data and insert text at various locations;
- compose text to be used in various situations;
- adjust the appearance of counters;
- adjust some other behaviour of the package.

We will highlight a few examples in section 3.2.

2.7 Exercise Styles

The package provides a mechanism to define exercise styles which customise the display of exercises in some coordinated fashion.

`\exercisestyle` Style(s) are activated by the command:

`\exercisestyle{`*styles*`}`

Here *styles* is a comma-separated list of styles, where each style is given by a pair *style*`[={`*argument*`}]`. The package defines a couple of standard styles:

- `solutionbelow=`*pos* (can take values `here`, `subproblem`, `subproblem*`, `problem`, `problem*`, `sheet` and `manual`; initially set to `subproblem`) – positions the solutions below the indicated environments; see section 2.2 for details.
- `problempointsat=`*pos* (can take values `start`, `start*`, `margin`, `end` and `manual`; initially set to `start`) – displays points in problems at the indicated location; see section 2.4 for details.
- `subproblempointsat=`*pos* (can take values `start`, `start*`, `margin`, `end` and `manual`; initially set to `end`) – displays points in subproblems at the indicated location; see section 2.4 for details.
- `solutionpointsat=`*pos* (can take values `start`, `start*`, `margin`, `end` and `manual`; initially set to `end`) – displays points in solutions at the indicated location; see section 2.4 for details.
- `problemby={`*counter*`}` – number problems with the prefix *counter*, i.e. reset the problem counter whenever *counter* increases and use a composite label *counter*`.`*problem* to identify problems.
- `equationby={`*counter*`}` – number the dedicated equation counters for sheets, problems and solutions with the prefix *counter*.
- `problembysheet` – number problems by sheet.
- `equationbysheet` – number dedicated equations for sheets, problems and solutions by sheet; note that the main equation counter is unaffected by this setting, it therefore makes sense to also activate the style `sheetequation` or use `\counterwithin{equation}{sheet}`.
- `pagebysheet` – number pages by sheet and denote pages by *sheet*`.`*page*; this style is useful to generate stable page numbers for a collection of sheets.
- `sheetequation[=true|false]` (no value implies `true`, initially set to `false`) – use a dedicated equation counter within sheets.
- `problemequation[=true|false]` (no value implies `true`, initially set to `false`) – use a dedicated equation counter within problems.

- `solutionequation`[`=true`|`false`] (no value implies `true`, initially set to `true`) – use a dedicated equation counter within solutions.
- `fracpoints`[`=true`|`false`] (no value implies `true`, initially set to `false`) – display fractional points for denominators 2, 4, 8; see section 2.4 for details.
- `twoside`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable two-sided layout; in two-sided layout, sheets will start on odd pages and empty pages are added at the end of sheets to produce an even number of pages.
- `subproblemdelimitem` – shift subproblem number delimiter ')' from subproblem label to subproblem item composition; needed if `\ref` to a subproblem `\label` should produce a bare identifier without a trailing ')'; assumes (and restores) numbering `\alph` and delimiter ')'.

`extstyle` Further exercise styles are defined by the package option `extstyle`, see section 2.8:

- `plainheader` – define a plain sheet header to display some essential exercise and sheet data: `course`, `institution`, `instructor`, `period` (optional), sheet `title`, see section 2.3; the line below the header, font styles and spaces can be adjusted, see the definition in section C.4.
- `contents` – display sheets and problems in the table of contents (as sections and subsections).
- `solutionsf` – display solutions in sans serif font family.
- `solutiondimproblem` – dim the problem text whenever solutions are displayed.
- `solutionsep` – separate the solutions from the remaining text by horizontal lines.

`\defexercisestyle` Custom styles can be defined by:

`\defexercisestyle{`*style*`}{`*init*`}`
`\defexercisestylearg[`*default*`]{`*style*`}{`*init*`}`

This feature can be used to predefine certain aspects of the exercises layout. For example, different default page layouts could be declared in this way. The first version declares a style which is initialised by the code *item* upon activation by `\exercisestyle{`*style*`[=true]}`. Note that `\exercisestyle{`*style*`=false}` does nothing. The second version declares a style which is activated by `\exercisestyle{`*style*`[={`*arg*`}]}` and which calls *item* with the argument `#1` referring to *arg* (or *default* if no argument is given).

2.8 Package Options

`\exercisesetup` Features and options of general nature can be selected by the commands:

`\usepackage[`*opts*`]{exframe}`
or `\PassOptionsToPackage{`*opts*`}{exframe}`
or `\exercisesetup{`*opts*`}`

`\PassOptionsToPackage` must be used before `\usepackage`; `\exercisesetup` must be used afterwards. *opts* is a comma-separated list of options.

The following options are available only when loading the package, i.e. they will not work within `\exercisesetup`:

- `extdata`[`=true`|`false`] (no value implies `true`, initially set to `false`) – define some more advanced metadata entries.

- `extstyle`[`=true`|`false`] (no value implies `true`, initially set to `false`) – define some more advanced styles.
- `metastr`[`=true`|`false`] (no value implies `true`, initially set to `false`) – load metastr package and use to handle pdf metadata and basic internationalisation, see section C.16 for details.
- `problemenv=`*name* – redefine environment name `problem`. This and the following alike options may be useful in quickly adjusting existing sources to the exframe framework if the original framework works similarly and no special features are used. Otherwise, it is highly advisable to leave the names of environments and counters defined by the package untouched.
- `subproblemenv=`*name* – redefine environment name `subproblem`.
- `solutionenv=`*name* – redefine environment name `solution`.
- `sheetenv=`*name* – redefine environment name `sheet`.
- `problemcounter=`*name* – redefine counter name `problem`.
- `subproblemcounter=`*name* – redefine counter name `subproblem`.
- `solutioncounter=`*name* – redefine counter name `solution`.
- `sheetcounter=`*name* – redefine counter name `sheet`.

The following options can be specified by all three methods described above:

- `solutions`[`=true`|`false`] (no value implies `true`, initially set to `false`) – Enable/disable display of solutions. Sets the conditional `\ifsolutions` accordingly.
- `pdfdata`[`=auto`|`manual`|`sheet`|`off`] (no value implies `auto`, initially set to `auto`) – control writing most relevant metadata to pdf files; has no effect without package hyperref.
- `lineno`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable writing of line numbers as comments into solution files.
- `twoside`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable two-sided layout; see section 2.7 for details.
- `solutionhref`[`=true`|`true`] (no value implies `true`, initially set to `false`) – enable/disable use of hyper-references from solutions to the corresponding problems; has no effect without package hyperref.
- `warntext`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable writing of relevant warning messages (points mismatch, point sums require update) into the document output for easier detection.
- `autolabelsheet`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable automatically assigning labels (`sheet:\sheettag`; can be adjusted) to sheets according to their tag `\sheettag`.
- `autolabelproblem`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable automatically assigning labels (`prob:\problemtag`; can be adjusted) to problems according to their tag `\problemtag`.
- `solutionbuf`[`=true`|`false`] (no value implies `true`, initially set to `true`) – enable/disable buffering for `solution` environments in order to control their display; disabling buffering can be helpful in debugging faulty `solution` environments; it might also resolve some tokenisation issues in special circumstances; note that the display of solutions cannot be suppressed with `\exercisesetup{solutions=false}` when buffering if disabled.
- `problembuf`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable buffering for `problem` environments in order to control their display.

- `subproblembuf`[`=true`|`false`] (no value implies `true`, initially set to `false`) – enable/disable buffering for `subproblem` environments in order to control their display.
- `emptytestchar=`*char* (initially set to '`&`') – character to use for testing whether an argument `#1` is empty via `\if&#1&...\fi`; if '`&`' causes trouble within tables, could try '`@`' instead.

3 Customisations

This section attempts to shed some light on the rich array of options for layout customisations. It is not at all complete, but it provides an overview and illustrates selected aspects by means of guidance and examples.

3.1 Guidance

The following provides some guidance on customisation options which might otherwise be hard to follow. Suggestions for topics to be expanded are welcome.

Package Philosophy. The philosophy of the present package is to define a low-level framework to describe exercises with solutions to be used in various situations. The aim is to provide the means to describe the content (problems, solutions, sheets) in a simple fashion and separate it from the various layout definitions and choices which will define the appearance of the content. The package itself provides a functional basic layout rather than an elaborate or particularly appealing one. However, it provides means to adjust the basic layout in many ways and to predefine custom layout schemes. The package is meant to collaborate with other packages which provide solutions for particular applications in order to produce an appealing layout for the given content.

Framework for Layout Customisation. The framework for defining the layout consists of an internal and a public component which allow for minor adjustments and major changes both with reasonable efforts.

The internal component implements some very coarse choices for the presentation of the content. It is hard-coded in the package (using internal macros starting with `exf@`) and its operation can be adjusted mainly through the available package options, see section 2.8, and through configuration switches for individual blocks.

The public framework uses a hierarchy of configuration macros defined by `\exerciseconfig`, see section 2.6. The internal component interfaces with the public framework through a set of higher-level interface macros. These include macros which produce the desired text in a well-prescribed situation as well as macros which perform some LaTeX operations such as adjusting presentation styles or modifying some variables. In many cases the structural higher-level macros have been set up to branch out to further lower-level macros prescribing presentation details.

The hierarchy has the following structure: High-level macros compose certain abstract elements into readable text, e.g. an intro/outro for some block of text made from space, decorations and abstract text elements. Medium-level macros compose elementary objects into abstract text elements, e.g. a header for some block of text composed from a label and a title with some space and delimiters in between. Low-level macros define elementary objects, such as delimiter characters, spaces, widths and words (translations).

The public framework has been set up such that elements at high, medium and low levels can be adjusted independently to arrive at a specific effect. For example, the composition of

problem titles can be adjusted abstractly, but also the representation of a specific delimiter can be tweaked from '.' to ':'; either change can be obtained by adjusting a single definition. This flexibility comes along with an unfortunate complexity to allow for adjustments to be implemented individually but also universally. Nevertheless, the implementation of the public configuration macros for a particular setup can be rather simple because the particular setup will neither use all of the available coarse layout choices nor all of the customisation options. For example, a higher-level macro for a problem title may directly produce the desired text without branching into further lower-level definitions. Also, only those higher-level macros which actually come to use in a particular setup will need to be implemented.

The following paragraphs provide guidance on aspects of the public framework. There is no complete manual for the framework, but a skilled LaTeX tinkerer can find further information and pieces of code ready for tweaking in the implementation notes in section C.3 and section C.4.

Labels for Problems and Sheets. The package provides many methods with various names to customise labels for problem and solution blocks. These names follow certain patterns (but are sometimes stuck with odd assignments due to backward compatibility). The following guide describes which methods come to use in which situations.

The labels can be grouped into two categories, items and titles, and the corresponding methods are typically called `...item...` and `...title...`. Items are enumerative identifiers for the [sub]problems such as '1.', 'b)' or '3iv)'. Titles serve as headers for [sub]problem or solution blocks which usually take the form "Problem 1", "2. Fermat's Last" or simply "c)" for a subproblem. Titles may use the corresponding item, and display it together with the type, name or using a different punctuation depending on the situation.

`composetitleproblem` `composenamedproblem` `composebareproblem`
The composition of problem titles through `composetitleproblem` depends mainly on whether an individual name is provided (`title` option to `problem` environment). The method passes on to `composenamedproblem` if a name is provided (second argument non-empty), otherwise to `composebareproblem`. It might also produce a plain name if no item is provided (first argument empty).

`osenamedproblemdelim` `mposenamedproblemsep` `posebareproblemdelim` `omposebareproblemsep` `poseitemproblemdelim` `omposeitemproblemsep`
Titles and items are separated by some space (`...sep`) and/or delimiting characters (`...delim`). The item within a named title is separated from the following name by `composenamedproblemdelim` and `composenamedproblemsep`. In an unnamed title, the type is separated from the following item by `composebareproblemsep`, and the item is followed by `composebareproblemdelim`. A plain item not used within a title is delimited by `composeitemproblemdelim`. The presentation of all titles and items is followed by `composeitemproblemsep`.

`composetocproblem`
Problem titles can also be made to appear in the table of contents, their display is governed by `composetocproblem` which distinguishes between named and unnamed problems. The method uses the above delimiter `composenamedproblemdelim` but no other customisations because detailed layout will be less welcome in the table of contents.

`composetitlesheet` `composetocsheet` `composemetasheet`
Sheet titles generated by `composetitlesheet` are similar to problem titles but simpler. They may or may not have a name, but there are no pre-defined delimiters. The two similar methods `composetocsheet` and `composemetasheet` govern the display of sheet titles in the table of contents and in pdf metadata, respectively.

`omposeitemsubproblem` `mposetitlesubproblem` `eitemsubproblemdelim` `oseitemsubproblemsep` `countersubproblem` `subproblemdelimitem`
Subproblem items and titles are simple as well because subproblems have no names. They are generated by `composeitemsubproblem` and `composetitlesubproblem`, respectively. The subproblem delimiter and separator are determined by `composeitemsubproblemdelim` and `composeitemsubproblemsep`, respectively. Note that the delimiter ')' within 'b)' may be declared as either `composeitemsubproblemdelim` or as part of the subproblem counter defined in `countersubproblem`. This behaviour is governed by the style `subproblemdelimitem`.

Labels for Solutions. The display of appropriate solution items and titles is very complex as it depends on four factors: (i) The first factor is whether solutions are displayed as an itemised list or as a plain list. (ii) The second factor is whether solutions are displayed individually, grouped by problem or assembled as a list for multiple problems. (iii) The third factor is whether the solution corresponds to a problem or subproblem. (iv) A fourth factor is whether a solution is the first one within a problem. In addition, the list of solutions may have a title (v) and sections (vi). Let us discuss the relevant combinations.

omposetitlesolutions
lutionsproblemsingle
olutionsproblemmulti

The solution list title ("Solutions") (v) generated by `composetitlesolutions` only applies to lists of (potentially) multiple problems (ii). The solution section title (vi) generated by `composetitlesolutionsproblem...` indicates the corresponding problem. It comes in two variants `...single` and `...multi` which distinguish a single problem ("Solution") from a list of (potentially) multiple problems ("Problem 2") (ii).

skipsolutionitem
skipsolutionitemsub
composeitemsolution
mposeitemsolutionsub
oseitemsolutionfirst
itemsolutionfirstsub

An itemised list of solutions (i) is invoked by a non-zero width `skipsolutionitem`[`sub`] (iii) for horizontal alignment of the solution text. In this case, solution items are displayed to the left of the line of horizontal alignment. These are generated by the configuration `composeitemsolution`[`sub`] (iii) irrespectively of the display mode (ii). By default, the current problem item is displayed as a section title (vi), therefore the solution items will only display the corresponding subproblem item leaving blank the solution items for problems. However, in an adjusted scheme without section titles (vi), the very first solution within a problem (iv) might additionally indicate the corresponding problem item `composeitemsolutionfirst`[`sub`] (iii).

etitlesolutionsingle
oseitemsolutiondelim

An individual solution (ii) formally part of a plain list (i) starts with the title "Solution:" which is generated by the method `composetitlesolutionsingle`. The delimiter ':' is defined by `composeitemsolutiondelim`.

setitlesolutionmulti
titlesolutionproblem
lesolutionsubproblem

The title for a grouped solution (ii) within a plain list (i) is generated by the method `composetitlesolutionmulti` which forwards to `composetitlesolution`[`sub`]`problem` (iii). Again, by default, these methods only display the subproblem item because the problem item is provided as a section title (vi) instead.

composetocsolution
composetocsolutions

Items for the table of contents corresponding to solution lists and problem solutions are generated by `composetocsolution`[`s`].

3.2 Examples

This section contains some sample customisations of the default (plain) layout which can be realised with moderate effort using the various tools provided by the package. Additional applications are shown in the sample files `exfsamp.tex` and `exfserm.tex`. Suggestions for further applications are welcome.

insertsheettitle

Sheet Title. An important setting is:

`\exerciseconfig{insertsheettitle}{`*code*`}`

The code *code* is meant to print the title or header of an exercise sheet. The minimalistic default code `\centerline{\getsheetdata{title}}` merely prints the sheet title "Sheet #" at the centre of a line. Commonly, one would replace this by a more elaborate header (potentially with some more information, appealing layout, logos, ...). In order to design a header template, it makes sense to retrieve data via `\getexercisedata` and `\getsheetdata` described in section 2.3. Likewise `\exercisedataempty` and `\sheetdataempty` can be used to display default values or alternative data if some particular data is not provided. An example is given by the `plainheader` extended style option defined in section C.4.

composetitleproblem composenamedproblem composebareproblem

Problem Title. Another noteworthy example is `composetitleproblem` to compose the title for a problem. It takes two parameters, the number and the title. The (somewhat simplified) default declarations are:

```
\exerciseconfig{composetitleproblem}[2]{\exerciseifempty{#2}
  {\getexerciseconfig{composebareproblem}{#1}}
  {\getexerciseconfig{composenamedproblem}{#1}{#2}}}
\exerciseconfig{composebareproblem}[1]{\getexerciseconfig{termproblem}
  #1\getexerciseconfig{composebareproblemdelim}}
\exerciseconfig{composenamedproblem}[2]
  {#1\getexerciseconfig{composenamedproblemdelim} #2}
```

This checks whether the title is empty. If no title is given use the configuration `composebareproblem` to compose "Problem #", where the term "Problem" is made abstract by the configuration `termproblem` (e.g. to support internationalisation). If a title is given use the configuration `composenamedproblem` to compose "#. *title*".

\exerciseifempty \exerciseifnotempty

Conditionals. Handy conditional commands to check whether an expression *expr* is empty are:

```
\exerciseifempty{expr}{empty code}{filled code}
\exerciseifnotempty{expr}{filled code}
```

Their main purpose is to test whether some provided expression *expt* is empty. They expand to the common TEX constructs `\if&#1&#2\else#3\fi` and `\if&#1&\else#2\fi` which work assuming that *expr* is not too exotic (e.g. it should not start with the character '`&`' and other special TEX characters or macros are potentially dangerous; also using this within tables can be troublesome; the character '`&`' can be reconfigured by the package option `emptytestchar`).

nsertsolutionsbefore insertsolutionsafter posetitlesolution...

Decorate Solutions with Boxes. The following adjustment changes the presentation of solutions to be surrounded by boxes with background colour (thanks to Till Bargheer for the example). This yields an alternative presentation that visually distinguishes problem from solution content.

The adjustment uses the package tcolorbox to supply such boxes. The coloured box environment `tcolorbox` is invoked by the customisations `insertsolutionsbefore` and `...after` which are called just before and after a block of solutions is:

```
\RequirePackage{tcolorbox}
\tcbuselibrary{breakable}
\exerciseconfig{insertsolutionsbefore}{%
  \begin{tcolorbox}[breakable,boxrule=0.25pt]}
\exerciseconfig{insertsolutionsafter}{\end{tcolorbox}}
\exerciseconfig{composetitlesolutionsingle}[2]{}
```

With the highlighting of solutions by boxes, the default introductory statement "Solution:" (when `solutionbelow` is `here` or `subproblem`) of a solution block becomes obsolete. This is achieved by removing the content of `composetitlesolutionsingle` (or similar composition definitions if `solutionbelow` is set otherwise).

solutionpointsat insertpointsmargin

Note that displaying points in the margin would have to be adjusted because the plain LATEX command `\marginpar` does not work inside a `tcolorbox` (due to nested floats). If this combination is desired, one should adjust the placement of margin paragraphs to the package marginnote:

```
\exercisestyle{solutionpointsat=margin}
\RequirePackage{marginnote}
\exerciseconfig{insertpointsmargin}[1]{\marginnote{\footnotesize#1}}
```

4 Information

4.1 Copyright

This work consists of the files `README.txt`, `exframe.ins` and `exframe.dtx` as well as the derived files `exframe.sty`, `exfsamp.tex`, `exfser`x`.tex` (x=`m`, `e`, `f`, `01`, `02`, `03`, `aa`), `exfsermk.sh`, `exfsermk.mak` and `exframe.pdf`.

4.2 Files and Installation

The package consists of the files:

`README.txt`	readme file
`exframe.ins`	installation file
`exframe.dtx`	source file
`exframe.sty`	package file
`exfsamp.tex`	sample file
`exfserm.tex`	multipart sample main file
`exfser0`n`.tex`	multipart sample sheet n=`1`, `2`, `3`
`exfseraa.tex`	multipart sample unused problems
`exfserp`x`.tex`	multipart sample problem x=`e`, `f`
`exfsermk.sh`	multipart sample compile script
`exfsermk.mak`	multipart sample makefile
`exframe.pdf`	manual

The distribution consists of the files `README.txt`, `exframe.ins` and `exframe.dtx`.

- Run (pdf)LaTeX on `exframe.dtx` to compile the manual `exframe.pdf` (this file).
- Run LaTeX on `exframe.ins` to create the package `exframe.sty` and the samples consisting of `exfsamp.tex`, `exfser`x`.tex` (x=`m`, `e`, `f`, `01`, `02`, `03`, `aa`), `exfsermk.sh`, `exfsermk.mak`. Copy the file `exframe.sty` to an appropriate directory of your LaTeX distribution, e.g. *texmf-root*`/tex/latex/exframe`.

4.3 Related Packages

The package makes use of other packages available at CTAN:

- This package relies on some functionality of the package verbatim to read verbatim code from the LaTeX source without expansion of macros. Compatibility with the verbatim package has been tested with v1.5q (2014/10/28).

- This package uses the package xkeyval to process the options for the package, environments and macros. Compatibility with the xkeyval package has been tested with v2.7a (2014/12/03).
- This package can use the package hyperref to include hyperlinks between problems and solutions. Compatibility with the hyperref package has been tested with v6.88e (2018/11/30).
- This package can use the package amstext (which is automatically loaded by amsmath) to display text within equations. Compatibility with the amstext package has been tested with v2.01 (2000/06/29).
- This package uses the command `\currfilename` provided by the package currfile (if available and loaded) to indicate the LaTeX source file in the generated metapost file. Compatibility with the currfile package has been tested with v0.7c (2015/04/23).
- This package can use the package metastr to write pdf metadata and to handle basic translations. Compatibility with the metastr package has been tested with v1.0 (2020/02/06).

There are several other LaTeX packages which offer a similar functionality varying largely in scope and sophistication:

- The package exsheets and its successor xsim provide a LaTeX 3 style for typesetting exercises with solutions. They offer options to hide or delay solutions, print only specific problems, deal with points, specify metadata, handle exercise collections, as well as some more specific options. They allow to adjust the layout and choose among predefined ones.
- The package exercise provides a style for typesetting exercises with solutions. It offers many options to hide or delay solutions, print only specific problems, specify some metadata as well as some more specific options. It allows to customise the layout.
- The package exercises provides a style for typesetting exercises with solutions. It offers options to hide solutions and deal with points. It allows basic customisation of the layout.
- The package exam provides a document class for typesetting exams conveniently. It offers many options to hide solutions, deal with points and deal with other exam-specific tasks. It allows to adjust the layout and choose among predefined ones.
- The package probsoln provides a style for typesetting exercises with solutions which are stored in a collection. It offers options to hide solutions and to assemble problems from an external collection.
- The packages uebungsblatt, uassign, mathexam, exsol, homework, jhwhw provide basic functionality for somewhat more particular situations.

See CTAN categories exercise and exam for further up-to-date packages.

The philosophy of the present package is to define a low-level framework to describe exercises with solutions to be used in various situations. The aim is to provide the means to describe the content (problems, solutions, sheets) in a simple fashion and separate it from the various layout definitions and choices which will define the appearance of the content. The interface was designed to reduce potential conflict with other packages and definitions. The package itself provides a functional basic layout rather than an elaborate or particularly appealing one. However, it provides means to adjust the basic layout in many ways and to predefine custom layout schemes. The package offers most of the functionality of the above packages, but (presently) misses out on some more advanced features. See section 3 on customisation assistance and examples and section 4.4 for feature suggestions.

4.4 Feature Suggestions

The following is a list of features which may be useful for future versions of this package:

- Define structures for multiple-choice questions.
- Add more guidance to section 3.1 on customisation settings.
- Add more examples of customisation settings to achieve specific goals to section 3.2. Please send suggestions.
- Option to hide problem text while maintaining access to embedded solutions (for a version containing only solutions): this is difficult to implement because the problem environment cannot simply be discarded, but would have to be scanned very carefully for the embedded solution; instead process problems to some document and save solutions to file, then read solutions from different document.

4.5 Revision History

v3.5: 2024/10/18

- more convenient delimiter configuration `compose...delim` for various items
- composition of problem titles in `composetitleproblem` split up further into the configurations `composebareproblem` and `composenamedproblem`; to enable delimiter '.' for problems without a name, use

```
\exerciseconfig{composebareproblemdelim}{.}
```

- style `subproblemdelimitem` added to shift delimiter ')' from subproblem reference labels to subproblem item composition.
- `composetitlesolutionsproblemmulti` improved (thanks to Daniel Wegmann for pointing out insufficient behaviour)
- option `forproblem` for `solution` to release association to previous subproblem
- option to `disable` particular subproblems and solutions including following `solution` block
- handling of first solutions within a problem block

v3.4: 2020/02/24

- lists of sheets, problems and subproblems; list iterators
- tags for subproblems, tag customisation
- interaction with package metastr
- minor fixes

v3.31: 2020/01/11

- `onlysolutions` environment for solution mode content
- sample multipart setup streamlined

v3.3: 2019/06/15

- control display of `problem` environments via package option `problembuf`: manual display, write to file, disable individual problems
- `solutionbelow` mode `here*` superseded by package option `solutionbuf`
- display total points within solution: `solutionpointsat` (thanks to Till Bargheer for suggestion)
- read points for current sheet, (sub)problem and solution (thanks to Johannes Hahn for suggestion)
- case switch for bonus points (thanks to Johannes Hahn for suggestion)
- option to `disable` particular problems, control by hook function (thanks to Manuel Benz for suggestion)
- provided interface `\showfracpoints` and `\exerciseconfig{frac}` for fractional points display
- filename extensions configurable

v3.2: 2019/05/01

- bonus points can be specified as `points=`[*regular*][`+`*bonus*]
- `solutionbelow` mode `here*` added for direct processing of the solution environment
- multipart sample added

v3.11: 2019/04/15

- fix interaction with package calc (thanks to Johannes Hahn for bug report)
- fix style `fracpoints` in combination with some [`sub`]`problempointsat` choices (thanks to Johannes Hahn for bug report)
- fix spacing for [`sub`]`problempointsat=margin` (thanks to Johannes Hahn for bug report)

v3.1: 2019/01/21

- alternate placement modes for solutions
- fixed expansion of problem title
- reset font size for problem text

v3.0: 2019/01/16

- renamed to `exframe.sty`
- first version published on CTAN
- overhaul and streamline interface
- solution processing remodelled
- changed metadata handling
- changed and generalised points handling
- generalised sectioning layout

- changed layout specification model
- insert hyperlinks using hyperref
- manual, example and installation package added

v2.0 – v2.6: 2014/10/03 – 2018/11/05

- changed metadata interface
- broadened scope
- added more layout options
- added more metadata
- added sheet and problem tags
- add and remember points

v1.1 – v1.6: 2014/08/07 – 2014/09/14

- renamed to `nbprob.sty`
- added metadata
- added points
- added layout configuration
- removed specific macros

v1.0 – v1.02: 2011/09/23 – 2013/03/17

- first version as `problems.cls`
- dedicated layout and macros for author's exercise sheets

A Standalone Sample

This section provides an example of how to use some of the exframe features. The resulting layout will be somewhat messy due to a random selection of features.

This example file describes a single exercise sheet. The other sheet of the series would be declared analogously in independent documents.

Preamble. Standard document class:

```
\documentclass[12pt]{article}
```

Use package geometry to set the page layout; adjust the paragraph shape:

```
\usepackage{geometry}
\geometry{layout=a4paper}
\geometry{paper=a4paper}
\geometry{margin=2.5cm}
\parindent0pt
\parskip0.5ex
```

Include amsmath, hyperref packages:

```
\usepackage{amsmath}
\usepackage{hyperref}
```

May include metastr package; below code adjusts to whether or not present:

```
\PassOptionsToPackage{loadlang=en|de}{metastr}
\PassOptionsToPackage{course=true}{metastr}
%%\usepackage{metastr}
%%\metasetlang{de}
```

Include exframe package:

```
\usepackage[extstyle]{exframe}
```

Solutions Switch. It will be useful to have the switch to turn on/off the display of solutions near the top of the source file, potentially with the opposite setting commented out:

```
\exercisesetup{solutions=false}
%%\exercisesetup{solutions=true}
```

Automatically put labels for (sub)problems:

```
\exercisesetup{autolabelproblem=true}
```

Layout Declarations. The following layout declarations adjust the general layout of exercise sheets. They may as well be moved into an include file.

Declare a header for exercise sheets to display several relevant pieces of data; display points total:

```
\exercisestyle{plainheader}
\exerciseconfig{composeheaderbelowright}{\getsheetdata{points}}%
```

Make the delimiter ‘)’ part of the subproblem item rather than the subproblem label:

```
\exercisestyle{subproblemdelimitem}
```

Redefine the appearance of some counters; sheets should be labelled by capital roman numerals, subproblems by lowercase roman numerals; declare the widest subproblem item to be expected:

```
\exerciseconfig{countersheet}{\Roman{sheet}}
\exerciseconfig{countersubproblem}{\roman{subproblem}}
\exerciseconfig{countersubproblemmax}{vii}
```

Automatically display an asterisk for all subproblems with bonus points only; remove space to separate items:

```
\exerciseconfig{insertsubprobleminfo}{%
 \switchpoints{}{\addprobleminfo*{%
   \hspace{-\getexerciseconfig{skipsubprobleminfo}}*}}%
 {}{}{\getsubproblempoints{}}}
```

Redefine the terms to be used for sheet(s); here, a German version:

```
\ifdefined\metaset
\metasetterm[en]{sheet}{Exercise Sheet}
\metasetterm[en]{sheets}{Exercise Sheets}
\metasetterm[de]{sheet}{\"Ubungsblatt}
```

```
\metasetterm[de]{sheets}{\"Ubungsbl\"atter}
\else
\exerciseconfig{termsheet}{\"Ubungsblatt}
\exerciseconfig{termsheets}{\"Ubungsbl\"atter}
\fi
```

Display points for problems in the margin; change margin display to use the left margin; use the abbreviated form '*n*p.':

```
\exercisestyle{problempointsat=margin}
\reversemarginpar
\exerciseconfig{composepointsmargin}[1]{#1p.}
\exerciseconfig{composepointspairmargin}[2]{%
  \ifdim#2pt=0pt#1p.%
  \else\ifdim#1pt=0pt+#2p.%
  \else#1+#2p.%
  \fi\fi}
```

Change the basic font style for all titles to be bold sans-serif:

```
\exerciseconfig{styletitle}{\sffamily\bfseries}
```

Add a significant amount of space below problems:

```
\exerciseconfig{skipproblembelow}{1.5cm}
```

Display half points as fractions:

```
\exercisestyle{fracpoints}
```

Show solutions below each problem (may try alternatives `subproblem` or `sheet`):

```
\exercisestyle{solutionbelow=problem}
```

Indent solutions for subproblems:

```
\exerciseconfig{skipsolutionitemsub}{-1pt}
```

Separate solutions by horizontal lines (extended style):

```
\exercisestyle{solutionsep}
```

Write metadata for sheet:

```
\exercisesetup{pdfdata=sheet}
```

Set title and author for pdf metadata; `title` is `course` plus `material` or sheet title; `author` is `instructor` or sheet author plus `institution`:

```
\ifdefined\metaset
\metaset[sep]{subtitle}{, }
\metaset{subtitle}{\ifsolutions\metatranslate[#1]{solutions} \fi%
  \metaif[use]{sheettitle}
   {\metapick[#1]{sheettitle}}
   {\metapick[#1]{material}}}
\metaset{author}{\exerciseifempty{\getsheetdata{author}}%
  {\metapick[#1]{instructor}}{\metapick[#1]{sheetauthor}},
  \metapick[#1]{institution}}
\else
\exerciseconfig{composemetasheet}[2]{\getexercisedata{course},
  \ifsolutions\getexerciseconfig{termsolutions} \fi%
  \exerciseifempty{#2}{\getexerciseconfig{termsheet} #1}{#2}}
\exercisedata{title=%
  {\getexercisedata{course},
```

```
  \ifsolutions\getexercisedata{solutions} \fi%
  \getexercisedata{material}}}
\exercisedata{author=%
  {\getexercisedata{instructor}, \getexercisedata{institution}}}
\fi
```

Exercise Series Data. Set some data on the current series:

```
\ifdefined\metaset
\metaset{institution}{Katharinen-Volksschule}
\metaset[de]{course}{Mathematik}
\metaset[en]{course}{Mathematics}
\metaset{instructor}{J.\ G.\ B\"uttner}
\metaset{period}{ca.\ 1786}
\metaset[de]{material}{\"Ubungsaufgaben}
\metaset[en]{material}{Exercise Problems}
\else
\exercisedata{institution={Katharinen-Volksschule}}
\exercisedata{course={Mathematik}}
\exercisedata{instructor={J.\ G.\ B\"uttner}}
\exercisedata{period={ca.\ 1786}}
\exercisedata{material={\"Ubungsaufgaben}}
\fi
```

Body.

```
\begin{document}
```

Start sheet number 5:

```
\begin{sheet}[number=5,label={sheet5}]
```

Start a problem with a title:

```
\begin{problem}[title={Sums},points=99+4]
```

Summary of points:

```
\exerciseloopstr{\getsubproblemlist{}}{c}%
\hfill\begin{tabular}{c|\exerciseloopret|c}
\exerciseloop{\getsubproblemlist{}}
 {&\ref{\getexerciseconfig{labelsubproblem}{#1}}}
&\ref{prob:\problemtag}\\\hline
\getexerciseconfig{termpoints}
\exerciseloop{\getsubproblemlist{}}{&\extractpoints{\getsubproblempoints{#1}}}
&\extractpoints{\getproblempoints{}}
\\
extra
\exerciseloop{\getsubproblemlist{}}{&\extractpoints*{\getsubproblempoints{#1}}}
&\extractpoints*{\getproblempoints{}}
\end{tabular}
```

Some introduction to the problem:

```
This problem deals with sums and series.
```

A subproblem with a local label:

```
\begin{subproblem}[points=2,difficulty=simple,label={\problemtag-simplesum}]
Compute the sum
\showpoints
```

```
\begin{equation}
1+2+3.
\end{equation}
```

Provide a solution for the subproblem (within the subproblem environment):

```
\begin{solution}
The result is
\begin{equation}
1+2+3=6.
\end{equation}
\end{solution}
```

End subproblem:

```
\end{subproblem}
```

Another subproblem:

```
\begin{subproblem}[points=97+0.5,difficulty=lengthy]
Compute the sum
\begin{equation}
1+2+3+\ldots+98+99+100.
\end{equation}
Keep calm and calculate!
%%That ought to keep him occupied for a while
\end{subproblem}
```

Provide a solution for the previous subproblem (layout may differ slightly from declaration within); declare author:

```
\begin{solution}[author={C.\ F.\ Gau\ss}]
We use the result $1+2+3=6$ from part \ref{\problemtag-simplesum})
to jumpstart the calculation. The remaining sums yield
\awardpoints*[1 for each remaining sum]{97}
\begin{equation}
6+4+5+\ldots+99+100=5050.
\end{equation}
Alternatively the summands can be grouped into pairs as follows:
\begin{align}
1+100&=101,\\
2+99&=101,\\
3+98&=101,\\
\ldots &\nonumber\\
50+51&=101.
\end{align}
These amount to 50 times the same number 101.
Therefore the sum equals
\begin{equation}
1+2+\ldots+99+100=50\cdot 101=5050.
\end{equation}
\textit{Ligget se!} \awardpoints{97+0.5}
\end{solution}
```

Some text between subproblems:

```
You may give the final part a try:
```

Final subproblem; this one is optional:

```
\begin{subproblem}[optional={optional},
  difficulty={requires inspiration},points={+3.5}]
```

```
Compute the series
\showpoints
\begin{equation}
1+2+3+\ldots
\end{equation}
```

Provide a solution:

```
\begin{solution}
The series is divergent, so the result is $\infty$ \awardpoints{+1}.
\par
However, after subtracting the divergent part,
the result clearly is
\begin{equation}
\zeta(-1)=-\frac{1}{12}\,,
\end{equation}
where the zeta-function $\zeta(s)$ is defined by
\begin{equation}
\zeta(s):=\sum_{k=1}^\infty \frac{1}{k^s}\,.
\end{equation}
This definition holds only for $s>1$ where the sum is convergent,
but one can continue the complex analytic function to $s<0$
\awardpoints{+1.5}.
\par
Another way of understanding the result
is to use the indefinite summation formula
for arbitrary exponent $s$ in the summand
(which also follows from the Euler--MacLaurin formula)
\begin{equation}
\sum_n n^s
= \frac{n^{s+1}}{s+1}
 -\sum_{j=0}^s  \frac{\zeta(j-s)\,s!}{(s-j)!\,j!}\,n^j
= \ldots - \zeta(-s)\,n^0.
\end{equation}
Curiously, the constant term with $j=0$ is just the desired result
but with the wrong sign
(in fact, the constant term of an indefinite sum is ambiguous;
for the claim we merely set $j=0$
in the expression which holds for others values of $j$)
\awardpoints{+0.5}.
In order to understand the sign,
we propose that the above formula describes the regularised result
for the sum with limits $+\infty$ and $n$
\begin{equation}
\sum_{k=+\infty}^n k^s
\simeq \frac{n^{s+1}}{s+1}
 -\sum_{j=0}^s  \frac{\zeta(j-s)\,s!}{(s-j)!\,j!}\,n^j.
\end{equation}
Then we flip the summation limits of the desired sum
to bring it into the above form
\awardpoints{+0.5}
\begin{equation}
\sum_{k=1}^\infty k^s
= -\sum_{k=\infty}^0 k^s
\simeq \zeta(-s).
\end{equation}
\end{solution}
```

End subproblem:

```
\end{subproblem}
```

End problem:

```
\end{problem}
```

Another problem; this one is untitled:

```
\begin{problem}[points=1, difficulty=insane]
Show that the equation
\begin{equation}
a^3+b^3=c^3
\end{equation}
has no positive integer solutions.
\end{problem}
```

A solution can also follow a problem (but the layout may be slightly different, e.g. here the space below the problem will appear before the solution):

```
\begin{solution}
\normalmarginpar
This is beyond the scope of this example.
\marginpar{\footnotesize\raggedright does not fit here.\par}
\end{solution}
```

Summary of points per problem and per subproblem for grading:

```
\ifsolutions\else
\textbf{Grading:}\par
\exerciseloopstr{\getproblemlist{}}{|c}
\begin{tabular}{|c|\exerciseloopret|c|}\hline
\getexerciseconfig{termsheet} \ref{sheet5}
\exerciseloop{\getproblemlist{*}}
 {&\ref{\getexerciseconfig{labelproblem}{#1}}}
&total
\\\hline
value
\exerciseloop{\getproblemlist{*}}
 {&\extractpoints{\getproblempoints{#1}}}%
&\extractpoints{\getsheetpoints{}}
\\\hline
\exerciseloop{\getproblemlist{*}}{&}
&\\\hline
\end{tabular}\qquad
\exerciseloop{\getproblemlist{*}}{
 \exerciseloopstr{\getsubproblemlist{#1}}{|c}
 \ifnum\value{exerciseloop}>0\relax
  \begin{tabular}{|c|\exerciseloopret|c|}\hline
  \getexerciseconfig{termproblem} \ref{\getexerciseconfig{labelproblem}{#1}}
  \exerciseloop{\getsubproblemlist{#1}}
   {&\ref{\getexerciseconfig{labelsubproblem}{##1}}}
  &total
  \\\hline
  value
  \exerciseloop{\getsubproblemlist{#1}}
   {&\extractpoints{\getsubproblempoints{##1}}}%
  &\extractpoints{\getproblempoints{#1}}
  \\\hline
  \exerciseloop{\getsubproblemlist{#1}}{&}
  &\\\hline
  \end{tabular}\quad
```

```
\fi
}
\fi
```

End sheet:

```
\end{sheet}
```

End of document body:

```
\end{document}
```

B Multipart Sample

The second example describes a series of exercise sheets which can be compiled as a collection or as individual sheets. This example describes a versatile setup with several convenient features; most of these features can be adjusted or removed easily as they mostly enhance the setup and do not interact with each other strongly.

B.1 Main File

The main source file is called `exfserm.tex`. It is referenced at several places within the setup, and when changing the name they need to be adjusted accordingly.

childdoc Mechanism. The setup uses the package childdoc to allow compilation of the series as a whole or in parts and with various sets of options:

```
\input{childdoc.def}
\childdocmain{}
```

Compilation Switches. Define compilation switches and declare their default settings. `\printsol` controls whether solutions should be printed or not; by default solutions are activated for compilation of a part, but not for the complete document. `\draftver` controls whether the final version of the document is to be compiled; concretely this affects the compilations of metapost figures, see below:

```
\ifchilddoc
\providecommand{\printsol}{y}
\else
\providecommand{\printsol}{n}
\fi
\providecommand{\draftver}{y}
\newif\ifdraft\if\draftver y\drafttrue\else\draftfalse\fi
```

Preamble. Standard document class:

```
\documentclass[12pt]{article}
```

```
% \ctanpkg{graphicx} package to display license logo:
\RequirePackage{graphicx}
```

hyperref Package. Use the hyperref package. Declare some options, e.g. use bookmarks only for complete document:

```
\PassOptionsToPackage{bookmarks=\ifchilddoc false\else true\fi}{hyperref}
\PassOptionsToPackage{bookmarksopen=true}{hyperref}
\RequirePackage{hyperref}
```

metastr Package. Use the metastr package to handle metadata and copyright information; disable usage of hyperxmp package if not installed; write auxiliary pdf metadata and rights information:

```
\ifdraft
\PassOptionsToPackage{draft}{metastr}
\fi
\IfFileExists{hyperxmp.sty}{}{\PassOptionsToPackage{hyperxmp=false}{metastr}}
\RequirePackage[course]{metastr}
\metaset[aux]{writepdf}{}
\metaset[rights]{writepdf}{}
```

exframe Package. Invoke exframe with extended data and styles:

```
\RequirePackage[extdata,extstyle]{exframe}
```

Set solutions switch and declare two-sided layout only if no solutions are printed; do not use solution buffer in draft mode for easier identification of potential compile errors:

```
\if\printsol n
\exercisesetup{solutions=false}
\exercisesetup{twoside=true}
\else
\exercisesetup{solutions=true}
\exercisesetup{twoside=false}
\ifdraft
\exercisesetup{solutionbuf=false}
\fi
\fi
```

Might want to display some metadata (only for partial compile):

```
%%\if\printsol n\else\showprobleminfo{author,source,recycle}\fi
```

Set some options. Automatically assign labels to problems. Include sheets and problems in table of contents. Separate solutions by horizontal rules. Count problems, equations and pages by sheet (unless compiling single problem). Collect solutions below each problem. Write pdf metadata for sheet when compiling single sheet:

```
\exercisesetup{autolabelproblem}
\exercisestyle{contents,solutionsep}
\ifchilddocmanual\else
\exercisestyle{pagebysheet,problembysheet,equationbysheet,sheetequation}
\fi
\exercisestyle{solutionbelow={problem}}
\ifchilddoc\ifchilddocmanual\else\exercisesetup{pdfdata=sheet}\fi\fi
```

Language. Redefine some terms:

```
\metasetlang{en-GB}

\metasetterm[en]{sheet}{sheet}
\metasetterm[en]{sheets}{sample sheets}
\metasetterm[en]{solution}{Solution}
\metasetterm[en]{solutions}{solutions}
```

Layout Definitions. Set page dimensions and layout:

```
\RequirePackage[a4paper,margin=2.5cm]{geometry}
\pagestyle{plain}
```

Remove paragraph indentation:

```
\setlength\parindent{0pt}
\setlength\parskip{\smallskipamount}
```

Show overfull lines:

```
\setlength\overfullrule{5pt}
```

Define turn page over mark; hide when printing solutions:

```
\newcommand{\turnover}{\ifsolutions\else\vfill%
  \hfill{\mathversion{bold}$\longrightarrow$}\newpage\fi}
```

Sheet Banner. Use standard sheet banner; show sheet `editdate` below banner (if declared); when compiling single sheet, display sheet due date instead:

```
\exercisestyle{plainheader}
\exerciseconfig{composeheaderbelowright}
 {\sheetdataempty{editdate}{}{version: \getsheetdata{editdate}}}
\ifchilddoc\ifsolutions\else
 \exerciseconfig{composeheaderbelowright}
  {\sheetdataempty{due}{}{due: \getsheetdata{due}}}
\fi\fi
```

Lorem. Define a macro `\lorem` to write out some paragraph of text for the example:

```
\def\lorem{Lorem ipsum dolor sit amet, consectetur adipisici elit,
  sed eiusmod tempor incidunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquid ex ea commodi consequat.
  Quis aute iure reprehenderit in voluptate velit esse
  cillum dolore eu fugiat nulla pariatur.
  Excepteur sint obcaecat cupiditat non proident,
  sunt in culpa qui officia deserunt mollit anim id est laborum.\par}
```

metapost Setup. Use package mpostinl (if available) to include some metapost figures within the source of the problems:

```
\IfFileExists{mpostinl.sty}{\RequirePackage{mpostinl}}{}
\ifdefined\mpostsetup
```

Setup mpostinl. Use checksums to invoke metapost only when figures change. Process all figures individually and immediately when in draft mode for child documents (avoids a second compilation pass). Number figures within sheet to provide a stable numbering upon insertion/deletion of new figures or partial compilation:

```
\mpostsetup{checksum}
\ifchilddoc\ifdraft\mpostsetup{now,nowall}\fi\fi
\ifchilddocmanual\else\mpostsetup{numberwithin={sheet}}\fi
```

Global metapost definitions. Define some latex macro to demonstrate usage of latex for typesetting labels. Define some global metapost variables, `paths` is an array of path variables, `xu` serves as a length unit to scale individual figures (use `interim xu:=...;` to set

scale for local figure only), `pensize` changes the size of the pen, `fillshape` fills and outlines a shape:

```
\mpostsetup{globaldef=true}
\begin{mposttex}
\def\figure{figure}
\end{mposttex}
\begin{mpostdef}
path paths[];
newinternal numeric xu;
xu:=1cm;
def pensize(expr s)=withpen pencircle scaled s enddef;
def fillshape(expr p,c)=
  fill p withcolor c;
  draw p
enddef;
\end{mpostdef}
\mpostsetup{globaldef=false}
```

Close optional mpostinl processing:

```
\fi
```

Document Data. Set document data for series of problem sheets:

```
\metaset{course}{exframe package samples}
\metaset{instructor}{N.\ Beisert}
\metaset{author}{Niklas Beisert, \metapick[#1]{institution}}
\metaset{institution}{exframe academy}
\metaset{period}{spring 2019}
\metaset{copyrightdate}{2019--2024}
```

For child documents declare part of:

```
\ifchilddoc
\metaset{partof}{\metatranslate[#1]{sheets} \metapick[#1]{course}}
\fi
```

Assemble some entries from given data:

```
\metaset[sep]{draft}{ -- }
\metaset[sep]{subtitle}{, }
\metaset{material}{\ifsolutions\metatranslate[#1]{solutions} \fi%
  \metatranslate[#1]{sheets}}
\exerciseconfig{composetitlesheet}[2]{\exerciseifempty{#2}%
  {\ifsolutions\metaterm{solutions}\else\metaterm{sheet}\fi\ #1}%
  {\ifsolutions\metaterm{solutions} \fi #2}}
\metaset{sheettitle}{\ifsolutions\metatranslate[#1]{solutions} \fi%
  \exerciseifempty{\getsheetdata{rawtitle}}%
   {\metatranslate[#1]{sheet} \thesheet}%
   {\getsheetdata{rawtitle}}}
\metaset{subject}{Lecture Series,
  \metapick[#1]{institution}, \metapick[#1]{period}}
```

License. It is good practice to specify a copyright line and a license for the document:

```
\metaset{copyrightowner}{\metapick[#1]{author}}
\ifchilddoc
\metacopyright{doc}
\else
```

```
\metacopyright{doc-parts}
\fi
\metaset{licenseprovider}{of \metapick[#1]{institution}}
\metalicense{consent}
```

Apply Creative Commons BY-SA license under certain conditions (no solutions, final version):

```
\ifsolutions\else\ifdraft\else
\metalicensecc{by-sa}
\fi\fi
```

Body. Start document body:

```
\begin{document}
```

Single Problem Display. The following code handles the compilation of individual problems from their own source file. Make sure to leave the conditional before issuing `\end{document}`:

```
\def\tmp{}
\ifchilddocmanual
\def\tmp{\end{document}}
\input{\childdocname}
\fi\tmp
```

Frontmatter. Do not print frontmatter for individual sheets. Define a plain page counter for frontmatter:

```
\setcounter{section}{-1}
\begingroup\ifchilddoc\else
\renewcommand{\thepage}{\arabic{page}}
```

Prepare a title page to display some relevant data:

```
\pdfbookmark[1]{\metaterm{title}}{title}
\thispagestyle{empty}
\vspace*{\fill}
\begin{center}
\metapick[course]{titletext}
\end{center}
\vspace*{\fill}\vspace*{\fill}
\newpage
```

Prepare a copyright and license page using data specified above:

```
\phantomsection\pdfbookmark[1]{\metaterm{copyright}}{copyright}
\thispagestyle{empty}
\vspace*{\fill}\vspace*{\fill}
\begin{center}\begin{minipage}{11cm}\raggedright
\metapick[print]{rightstext}
\end{minipage}\end{center}
\vspace*{\fill}\vspace*{\fill}\vspace*{\fill}
\newpage
```

Print table of contents:

```
\makeatletter\renewcommand\@pnumwidth{2.4em}\makeatother
\setcounter{tocdepth}{2}
```

```
\phantomsection\pdfbookmark[1]{\metaterm{contents}}{contents}
{\parskip0pt\tableofcontents}
\exercisecleardoublepage\setcounter{page}{1}
```

End of frontmatter:

```
\fi\endgroup
```

Include Sheets. Include problem sheets:

```
\include{exfser01}
\include{exfser02}
\include{exfser03}
```

Include sheet to collect unused problems (only process for individual sheets, i.e. itself):

```
\def\jobnameunused{exfseraa}
\ifx\childdocname\jobnameunused\include{\jobnameunused}\fi
```

End. End of document body:

```
\end{document}
```

B.2 Sheet File

Provide some source files `exfser01.tex`, `exfser02.tex`, `exfser03.tex`, `exfseraa.tex` for problem sheets.

childdoc Mechanism. Instruct the package childdoc to compile only the present sheet if source is compiled by latex:

```
%%\providecommand{\printsol}{n}
\input{childdoc.def}
\childdocof{exfserm}
```

The parameter of `\childdocof` must match the main file name `exfserm`. Uncommenting the commented line suppressed printing of the solution.

Sheet Environment. Declare a sheet with intended due date:

```
\begin{sheet}[due={2019-04-29}]
```

Adjust due date for each sheet. For sheet containing unused problems `exfseraa.tex`, declare a sheet `title={unused problems}` instead of due date.

Problems. Start a problem:

```
\begin{problem}[title={Sample A}]
```

Let us declare a figure using mpostinl (if available). Denote it by the label *tag*`-fig`, where *tag* is the tag of the problem (in order to avoid potential conflicts with other problems; *tag* is assigned automatically or by specifying the option `tag` for the `problem` environment). Also declare a figure *tag*`-solfig` for use within the solution:

```
\ifdefined\mpostuse
\begin{mpostfig}[label={\problemtag-fig}]
interim xu:=1.5cm;
```

```
paths[1]:=fullcircle scaled 1xu;
fillshape(paths[1], 0.7white) pensize(1pt);
label(btex \figure etex, center(paths[1]));
\end{mpostfig}
\begin{onlysolutions}
\begin{mpostfig}[label={\problemtag-solfig}]
interim xu:=1.5cm;
paths[1]:=fullcircle scaled 1xu;
paths[2]:=((subpath (-2,2) of paths[1])--cycle) shifted (+0.5xu,0);
paths[3]:=((subpath (2,6) of paths[1])--cycle) shifted (-0.5xu,0);
fillshape(paths[2], 0.7white) pensize(1pt);
fillshape(paths[3], 0.7white) pensize(1pt);
\end{mpostfig}
\end{onlysolutions}
\fi
```

Write a problem body with figure, some subproblems and a solution:

```
\lorem

\begin{subproblem}
\lorem
\begin{center}
\ifdefined\mpostuse\mpostuse{\problemtag-fig}\else figure\fi
\end{center}
\lorem
\end{subproblem}

\begin{solution}
\lorem
\begin{center}
\ifdefined\mpostuse\mpostuse{\problemtag-solfig}\else figure\fi
\end{center}
\lorem
\end{solution}

\begin{subproblem}
\lorem
\end{subproblem}

\begin{solution}
\lorem
\end{solution}

\lorem

\begin{subproblem}
\lorem
\end{subproblem}

\begin{solution}
\lorem
\end{solution}
```

End the problem:

```
\end{problem}
```

Start new page:

```
\turnover
```

Write a second problem to accompany the first one:

```
\begin{problem}
```

Problem body without a figure; this time the `solution` environments are included in the `subproblem` environments:

```
\lorem

\begin{subproblem}
\lorem
\begin{solution}
\lorem
\end{solution}
\end{subproblem}

\begin{subproblem}
\lorem
\begin{solution}
\lorem
\end{solution}
\end{subproblem}
```

End the problem:

```
\end{problem}
```

End Sheet. End the sheet:

```
\end{sheet}
```

B.3 Individual Problem Files

It may be more convenient to define each problem in an individual file, so that a sheet can be composed by including the appropriate problem files. In such a setup, the childdoc mechanism allows to compile each problem individually.

To that end, prepare a file `exfserp`*nn*`.tex` containing the `problem` environment. This file should start with:

```
%%\providecommand{\printsol}{n}
\input{childdoc.def}
\childdocby{exfserm}
```

and end with `\endinput`. Then compose the problem sheet by including the appropriate set of problem files via `\input{exfserp`*nn*`}`.

In the example, a sheet file `exfser03.tex` includes two problem files `exfserpe.tex` and `exfserpf.tex`.

B.4 Make Scripts

The setup allows the compilation in various modes for editing purposes. In order to generate and update a complete set of documents for distribution (all individual sheets and a collection of sheets; with and without solutions), it makes sense to use the software development utility `make`. The compilation of individual components is simplified by a `bash` shell script.

Compile Script. The `bash` shell script `exfsermk.sh` compiles one part of the collection of problem sheets in a given mode.

Shebang for `bash` script:

```
#!/bin/bash
```

Configure and declare variables with default values. `srcmain` defines the name of the main source file; `srcsec`*nn* defines the name of the sheet source file; `trglist` defines the target file names; `trgsol` defines the target modes; `sheets` is a list of allowable sheet identifiers *nn*:

```
srcmain="exfserm"
srcsec="exfser"
trglist=(Problems Solutions)
trgsol=(n y)
secnum="01 02 03 aa"
```

Display usage:

```
if [ -z $1 ]
then
  echo "Usage:
  $0 number [version]
    number: number of sheet, 0 for combined document
    version: 0 for problems, 1 for solutions
  $0 filename
    filename: target file to be compiled"
  exit 1
fi
```

Configure and declare variables with default values. `num` takes the sheet number; `ver` takes the compile mode;

```
num="$1"
ver="$2"
nl=$'\n'
secokay=""
make=".pdf"
```

Check if the parameter matches any of the acceptable output file names:

```
for v in "${trglist[@]}"
do
  if [[ $num =~ ^$v ]]
  then
    ver=$v
    num=${num#$v}
    if [[ $num =~ ^.*\.tex$ ]]; then make=".tex"; fi
    num=${num%%.*}
  fi
done
```

Ensure that `num` is a two-digit number, prepend '0' otherwise:

```
if [[ $num =~ ^[0-9]$ ]]; then num="0$num"; fi
if [[ $num == "00" ]]; then num=""; fi
```

Check whether `num` is acceptable:

```
if [[ -z $num ]]; then secokay="okay"; fi
for v in $secnum
do
```

```
534   if [[ "$num" == "$v" ]]; then secokay="okay"; fi
535 done
```

Otherwise display error message and exit:

```
536 if [[ -z $secokay ]]
537 then
538   echo "error: unknown sheet"
539   exit 1
540 fi
```

Disable newline character for command line tex code:

```
541 if [[ "$make" == ".pdf" ]]; then nl=""; fi
```

Function to compile a component. Set up childdoc mechanism according to desired component. Compile two passes, first in `-draftmode`. Suppress messages by `mpost`. Display warning messages in log file:

```
542 function docompile
543 {
544   if [[ -z $num ]]
545   then
546     job="$srcmain"
547     fwd="\\childdocforward{$srcmain}"
548   else
549     job="$srcsec$num"
550     fwd="\\childdocforward[$srcmain]{$srcsec$num}"
551   fi
552   body="\\def\\jobname{$job}$optdef\\input{childdoc.def}$fwd"
553   for pass in first main extra
554   do
555     par="";
556     if [[ "$pass" == "first" ]]; then par="-draftmode"; fi
557     drop="This is|entering extended mode|\\write18"
558     drop="$drop|Preloading the plain mem file|mpost\.mp|plain\.mp"
559     pdflatex -shell-escape -interaction=batchmode $par \
560       -jobname "$trg" "$body" | grep -vE "$drop"
561     if [[ "$pass" != "main" ]]; then continue; fi
562     if ! (grep -E -q "may have changed|rerunfilecheck Warning" "$trg.log")
563       then break; fi
564   done
565   grep -E "^! |Warning|Error|Undefined|Overfull|Underfull" "$trg.log"
566 }
```

Function to generate a childdoc compile file with specific options:

```
567 function writesource
568 {
569   if [[ -z $num ]]
570   then
571     fwd="\\childdocforward{$srcmain}"
572   else
573     fwd="\\childdocforwardprefix[$srcmain]{$target}{$srcsec}"
574   fi
575   body="$optdef\\input{childdoc.def}$nl$fwd"
576   echo "$body" > $trg.tex
577 }
```

Translate versions to parameter values, configure variables and select appropriate function for intended task:

```
for i in "${!trglist[@]}"
do
  if [[ -z $ver || "$ver" == "${trglist[$i]}" || $ver = $i ]]
  then
    target="${trglist[$i]}"
    sol="${trgsol[$i]}"
    trg="$target$num"
    optdef="\\def\\draftver{n}$nl\\def\\printsol{$sol}$nl"
    if [[ "$make" == ".pdf" ]]; then docompile; else writesource; fi
  fi
done
```

Finish with blank line:

```
echo
```

Makefile. Define a make file `exfsermk.mak` for the project. The compilation is then started by `make -f exfsermk.mak` .... More conveniently, in a single-project setup within the directory, this file would be called `Makefile` in which case it suffices to just run `make` ....

Configuration definitions:

```
SRCMAIN = exfserm
SRCSEC = exfser
SRCPRB = exfserp
SCRIPT = exfsermk.sh
MAKEFILE = exfsermk.mak
TRGLIST = Problems Solutions
SECNUM = 01 02 03 aa
PREREQS = $(SRCMAIN).tex

SRCSECFILES = $(SECNUM:%=$(SRCSEC)%.tex)
TRGMAINFILES = $(foreach trg,$(TRGLIST),$(trg).pdf)
TRGSECFILES = $(foreach trg,$(TRGLIST),$(trg).pdf $(SECNUM:%=$(trg)%.pdf))
GENFILES = $(foreach trg,$(TRGLIST),$(trg).tex $(SECNUM:%=$(trg)%.tex))
BAKFILES = $(PREREQS) $(SRCSECFILES) $(GENFILES)\
           $(MAKEFILE) $(SCRIPT) $(SRCPRB)*
```

Define some abstract targets; `default` is the default target when no parameters are given:

```
default: sheets ;
main: $(TRGMAINFILES) ;
sheets: $(TRGSECFILES) ;
sheet%: $(foreach trg,$(TRGLIST),$(trg)%.pdf) ;
all: main sheets ;
sources: $(GENFILES) ;
```

Compile particular files via `exfsermk.sh` bash script. Note that command lines have to start with a tab character (represented by 8 spaces here):

```
$(TRGMAINFILES): $(SRCSECFILES) $(PREREQS)
 bash ./$(SCRIPT) $@
$(word 1,$(TRGLIST))%.pdf: $(SRCSEC)%.tex $(PREREQS)
 bash ./$(SCRIPT) $@
$(word 2,$(TRGLIST))%.pdf: $(SRCSEC)%.tex $(PREREQS)
 bash ./$(SCRIPT) $@
$(GENFILES):
 bash ./$(SCRIPT) $@
```

Touch main file for recompile:

```
touch:
 touch $(SRCMAIN).tex
```

Define `clean` target to remove all intermediate compilation files:

```
clean:
 rm -f $(foreach ext,.aux .log,$(SECNUM:%=$(SRCSEC)%$(ext)))
 rm -f $(foreach trg,$(TRGLIST),$(SECNUM:%=$(trg)%.log) $(trg).log)
 rm -f $(foreach ext,.aux .log .out .toc,$(SRCMAIN)$(ext))
 rm -f $(foreach ext,.mp .mpx -*.mps,$(SRCMAIN)$(ext))
 rm -f $(foreach ext,-tmp.log -tmp.mp -tmp.mpx,$(SRCMAIN)$(ext))
 rm -f mpxerr.tex mpxerr.log mpxerr.dvi texput.log
 rm -f $(patsubst %,$(SRCPRB)*%,.aux .log .mp .mpx -*.mps)
 rm -f $(patsubst %,$(SRCPRB)*%,-tmp.log -tmp.mp -tmp.mpx)
```

Define `clean-bak` target to remove all backup files ending in '~' or '`.bak`':

```
clean-bak:
 rm -f $(BAKFILES:%=%~) $(BAKFILES:%=%.bak)
```

Define `clean-all` target to remove all generated files for a clean source directory:

```
clean-all:  clean
 rm -f $(TRGSECFILES) $(TRGMAINFILES) $(GENFILES)
 rm -f $(SECNUM:%=$(SRCSEC)%.pdf) $(SRCMAIN).pdf
 rm -f $(SRCPRB)*.pdf
```

C Implementation

This section describes the package `exframe.sty`.

C.1 General Definitions

Required Packages. The package loads the package verbatim and xkeyval if not yet present. verbatim is used for solution environment reading and xkeyval is used for extended options processing:

```
\RequirePackage{verbatim}
\RequirePackage{xkeyval}
```

General Definitions.

`\exf@empty` Define an empty macro for comparison by `\ifx`:

```
\def\exf@empty{}
```

`\exf@tmpdim` Define a length for temporary storage:

```
\newlength\exf@tmpdim
```

`\exf@exptwo` A macro to conveniently expand the third token in line:

```
\def\exf@exptwo#1{\expandafter#1\expandafter}
```

`\exf@exparg` A macro to conveniently expand the first token of an argument following arbitrary code:

```
\long\def\exf@expswitch#1#2{#2{#1}}
\long\def\exf@exparg#1#2{\exf@exptwo\exf@expswitch{#2}{#1}}
```

`\exf@csdo` `\exf@csdotwo` Some macros to conveniently expand `\csname` arguments before expanding the macro:

```
\def\exf@csdo#1#2{\expandafter#1\csname#2\endcsname}
\def\exf@csdotwo#1#2#3{\exf@exptwo#1#2\csname#3\endcsname}
```

`\exf@csor` A macro to return `\csname` or default if not exists:

```
\def\exf@csor#1#2{\ifcsname#1\endcsname\csname#1\endcsname\else#2\fi}
```

`\exf@append@def` `\exf@prepend@def` Add definitions to macros (after or before original content):

```
\long\def\exf@append@def#1#2{\exf@exptwo\def#1\expandafter{#1#2}}
\long\def\exf@prepend@switch#1#2#3{#2{#3#1}}
\long\def\exf@prepend@def#1#2{\exf@exptwo\exf@prepend@switch{#1}{\def#1}{#2}}
```

`\exf@expsetkeys` A version of `\setkeys` from xkeyval which expands first:

```
\newcommand{\exf@expsetkeys}[2]{\edef\exf@tmp{#2}%
  \exf@exparg{\setkeys{#1}}{\exf@tmp}}
```

`\exf@isif` Execute `#3` if content of macro `#1` equals `#2`:

```
\newcommand{\exf@ifis}[3]%
  {\def\exf@tmp{#2}\ifx#1\exf@tmp#3\fi}
```

`\exf@href` Display text with hyperreference passed by macro `#1` (in case hyperref is loaded and the reference is defined and not empty):

```
\newcommand{\exf@href}[2]{%
  \ifdefined#1\ifx#1\exf@empty#2\else%
   \ifdefined\hyperlink\protect\hyperlink{#1}{#2}\else#2\fi\fi\else#2\fi}
```

`\exf@text` `\exf@ensuretext` Two macros to display text in math mode. `exf@text` is a wrapper for `\text` of amstext in case the latter package is loaded. `exf@ensuretext` makes sure the text is set in text mode or within an `\mbox` in math math:

```
\newcommand{\exf@text}[1]{\ifdefined\text\text{#1}\else#1\fi}
\newcommand{\exf@ensuretext}[1]{\ifmmode\mbox{#1}\else#1\fi}
```

`\exf@addcontentsline` Add a line to the table of contents unless macro in argument `#1` is empty:

```
\newcommand{\exf@addcontentsline}[2]{%
  \ifx#1\exf@empty\else\addcontentsline{toc}{#1}{#2}\fi}
```

Auxiliary File Data Storage.

`\exf@notedata` Process stored data by handler:

```
\newcommand{\exf@notedata}[3]{\csname exf@notedata@#1\endcsname{#2}{#3}}
```

Make sure the macros in code written to the `.aux` file exist:

```
\AtBeginDocument{\immediate\write\@auxout{%
  \string\providecommand{\string\exf@notedata}[3]{}}}
```

`\exf@writedata` Write data to the `.aux` file:

```
\newcommand{\exf@writedata}[3]%
  {\immediate\write\@auxout{\string\exf@notedata{#1}{#2}{#3}}}
```

C.2 Package Setup

Initialisation Options.

exframe.sty Some setup options are available while loading the package only.

Configure names of main environments and counters:

```
\def\exf@problemname{problem}
\def\exf@subproblemname{sub\exf@problemname}
\def\exf@solutionname{solution}
\def\exf@sheetname{sheet}
\def\exf@problemcounter{problem}
\def\exf@subproblemcounter{sub\exf@problemcounter}
\def\exf@solutioncounter{solution}
\def\exf@sheetcounter{sheet}
\define@key{exframe.sty}{problemenv}{\def\exf@problemname{#1}}
\define@key{exframe.sty}{subproblemenv}{\def\exf@subproblemname{#1}}
\define@key{exframe.sty}{solutionenv}{\def\exf@solutionname{#1}}
\define@key{exframe.sty}{sheetenv}{\def\exf@sheetname{#1}}
\define@key{exframe.sty}{problemcounter}{\def\exf@problemcounter{#1}}
\define@key{exframe.sty}{subproblemcounter}{\def\exf@subproblemcounter{#1}}
\define@key{exframe.sty}{solutioncounter}{\def\exf@solutioncounter{#1}}
\define@key{exframe.sty}{sheetcounter}{\def\exf@sheetcounter{#1}}
```

Whether to provide some extended configuration options (available while loading only):

```
\define@boolkey{exframe.sty}[exf@]{extdata}[true]{}
\define@boolkey{exframe.sty}[exf@]{extstyle}[true]{}
```

Whether to load package metastr (available while loading only):

```
\define@boolkey{exframe.sty}[exf@]{metastr}[true]{}
```

Setup Options.

exf@setup All remaining setup options are available also when the package is already loaded.

Main switch for solutions:

```
\define@boolkey{exf@setup}[]{solutions}[true]{}
```

Switch for writing pdf metadata:

```
\define@choicekey{exf@setup}{pdfdata}%
  {auto,manual,sheet,off}[auto]{\def\exf@metadata{#1}}
\def\exf@metadata{auto}
```

Write line number indicators to output file:

```
\define@boolkey{exf@setup}[exf@]{lineno}[true]{}
```

Prepare two-sided sheets:

```
\define@boolkey{exf@setup}[exf@]{twoside}[true]{}
```

Generate hyperreferences from solutions to corresponding problems:

```
\define@boolkey{exf@setup}[exf@]{solutionhref}[true]{}
\exf@solutionhreftrue
```

Automatically generate labels for sheets and problems:

```
\define@boolkey{exf@setup}[exf@]{autolabelsheet}[true]{}
\define@boolkey{exf@setup}[exf@]{autolabelproblem}[true]{}
```

Write warning message to document for better detection of inconsistencies:

```
\define@boolkey{exf@setup}[exf@]{warntext}[true]{}
```

Activate buffering of solutions:

```
\define@boolkey{exf@setup}[exf@]{solutionbuf}[true]{}
\exf@solutionbuftrue
```

Activate buffering of problems:

```
\define@boolkey{exf@setup}[exf@]{problembuf}[true]{}
```

Activate buffering of subproblems:

```
\define@boolkey{exf@setup}[exf@]{subproblembuf}[true]{}
```

`\exf@emptytestchar` Redefine character for testing emptyness:

```
\def\exf@emptytestchar{&}
\define@key{exf@setup}{emptytestchar}{\def\exf@emptytestchar{#1}}
```

Processing. Process global options while loading package:

```
\ProcessOptionsX<exframe.sty,exf@setup>
```

`\exercisesetup` Configure package when package is already loaded:

```
\newcommand{\exercisesetup}[1]{\exf@expsetkeys{exf@setup}{#1}}
```

Additional Packages. Load metastr package if desired:

```
\ifexf@metastr
\PassOptionsToPackage{course=true}{metastr}
\RequirePackage{metastr}
\fi
```

Solutions Only Processing.

`onlysolutions` Process block only in solutions mode:

```
\newenvironment{onlysolutions}%
 {\ifsolutions\else%
   \let\endonlysolutions\endcomment%
   \expandafter\comment\fi}%
 {}
```

C.3 Configuration

This section defines and describes the various configuration options provided by the package. It also serves as a manual, and most code can be recycled and adjusted for individual configurations:

Definitions.

`\exerciseconfig` Set a configuration macro; store definition in `exf@config@#2`; use `\newcommand` for macros with arguments, but (non-long) `\def` for plain definitions:

```
\newcommand{\exerciseconfig}[1]{%
  \@ifnextchar[{\exf@configopt{#1}}{\exf@configoopt{#1}}}
\long\def\exf@configopt#1[#2]#3{%
  \exf@csdo\def{exf@config@#1}{}%
  \exf@csdo\renewcommand{exf@config@#1}[#2]{#3}}%
\long\def\exf@configoopt#1#2{\exf@csdo\def{exf@config@#1}{#2}}
```

exerciseconfigappend
xerciseconfigprepend

Append to a (parameterless) configuration macro:

```
\newcommand{\exerciseconfigappend}[2]{%
  \exf@csdo\exf@append@def{exf@config@#1}{#2}}
\newcommand{\exerciseconfigprepend}[2]{%
  \exf@csdo\exf@prepend@def{exf@config@#1}{#2}}
```

\getexerciseconfig

Get configuration macro:

```
\newcommand{\getexerciseconfig}[1]{\csname exf@config@#1\endcsname}
```

\exerciseconfigempty

Test whether configuration macro `#1` is empty; execute `#2` if empty, otherwise execute `#3`:

```
\newcommand{\exerciseconfigempty}[3]{\exf@csdo\ifx{exf@config@#1}\exf@empty%
  #2\else#3\fi}
```

\exerciseifemtpy
\exerciseifnotempty

Code to test whether `#1` (expanded) is empty; execute `#2` if empty, otherwise execute `#3`:

```
\long\def\exerciseifempty#1#2#3{%
  \if\exf@emptytestchar#1\exf@emptytestchar#2\else#3\fi}
\long\def\exerciseifnotempty#1#2{%
  \if\exf@emptytestchar#1\exf@emptytestchar\else#2\fi}
```

Terms.

term...

Terms for sheet, problem, solution and points (for adjustment or internationalisation):

```
\exerciseconfig{termsheet}{Sheet}
\exerciseconfig{termsheets}{Sheets}
\exerciseconfig{termproblem}{Problem}
\exerciseconfig{termproblems}{Problems}
\exerciseconfig{termsolution}{Solution}
\exerciseconfig{termsolutions}{Solutions}
\exerciseconfig{termpoint}{point}
\exerciseconfig{termpoints}{points}
```

Formatting Styles.

style...

Formatting styles to be applied for various parts of text. Different styles will be applied in sequence from more general to more specific.

Basic style for all exercise text:

```
\exerciseconfig{styletext}{\normalsize\normalfont}
```

Style for problems:

```
\exerciseconfig{styletextproblem}{}
```

Style for solutions:

```
\exerciseconfig{styletextsolution}{\footnotesize}
```

Basic style for titles:

```
\exerciseconfig{styletitle}{\bfseries}
```

Style for problem titles:

```
\exerciseconfig{styletitleproblem}{\large}
```

Style for subproblem titles:

```
\exerciseconfig{styletitlesubproblem}{}
```

Style for solution titles:

```
\exerciseconfig{styletitlesolution}{}
```

Style for problem section title in solution block:

```
\exerciseconfig{styletitlesolutionsproblem}{\small}
```

Style for solution block title:

```
\exerciseconfig{styletitlesolutions}{\normalsize}
```

Style for problem block title:

```
\exerciseconfig{styletitleproblems}{\Large}
```

Spacing.

`skip...` Spaces related to various elements. Vertical space is typically combined with space declared elsewhere using `\addvspace`.

Space above problem environment:

```
\exerciseconfig{skipproblemabove}{3.25ex plus 1ex minus 1.5ex}
```

Space below problem environment:

```
\exerciseconfig{skipproblembelow}{3pt plus 1pt minus 1pt}
```

Space below or after problem title; positive numbers generate vertical space (problem body is started in new paragraph), negative numbers generate horizontal space (problem body continues on opening line):

```
\exerciseconfig{skipproblemtitle}{3pt plus 1pt minus 1pt}
```

Horizontal space between items in the problem opening line:

```
\exerciseconfig{skipprobleminfo}{0.5em}
```

Space for problem item and indentation; `0pt` means no indentation and direct display of title; positive numbers define an absolute amount; `-1pt` (or any negative number) computes the amount of indentation from the width of (standard) item plus separator:

```
\exerciseconfig{skipproblemitem}{0pt}
```

Spaces related to subproblem environment; analogous to spaces related to problem environment, see above:

```
\exerciseconfig{skipsubproblemabove}{1.5ex plus 0.5ex minus 1ex}
\exerciseconfig{skipsubproblembelow}{1.5ex plus 0.5ex minus 1ex}
\exerciseconfig{skipsubproblemtitle}{-1em}
\exerciseconfig{skipsubprobleminfo}{0.25em}
\exerciseconfig{skipsubproblemitem}{-1pt}
```

Spaces related to solution environment; analogous to spaces related to problem environment,

```
\exerciseconfig{skipsolutionabove}{0ex}
\exerciseconfig{skipsolutionbelow}{1.5ex plus 0.5ex minus 1ex}
\exerciseconfig{skipsolutiontitle}{-0.5em}
\exerciseconfig{skipsolutioninfo}{0.25em}
```

`skipsolutionitem` and `skipsolutionitemsub` are analogous to `skipproblemitem` described above; they apply to solutions corresponding to problems and subproblems, respectively:

```
\exerciseconfig{skipsolutionitem}{0pt}
\exerciseconfig{skipsolutionitemsub}{0pt}
```

Spaces related to solution blocks; space above and below a solution block:

```
\exerciseconfig{skipsolutionsabove}{1.5ex plus 0.5ex minus 1ex}
\exerciseconfig{skipsolutionsbelow}{1.5ex plus 0.5ex minus 1ex}
```

Space above problem titles in a solution block:

```
\exerciseconfig{skipsolutionsproblemabove}{1.0ex plus 0ex minus 0.5ex}
```

Space following problem titles in a solution block (with legacy definition):

```
\exerciseconfig{skipsolutionsproblemtitle}{1.0ex plus 0ex minus 0.5ex}
\exerciseconfig{skipsolutionsproblem}{\exf@config@skipsolutionsproblemtitle}
```

Space following title of a solution block:

```
\exerciseconfig{skipsolutionstitle}{1.0ex plus 0ex minus 0.5ex}
```

Spaces related to problem blocks; space above and below a problem block:

```
\exerciseconfig{skipproblemsabove}{1.5ex plus 0.5ex minus 1ex}
\exerciseconfig{skipproblemsbelow}{1.5ex plus 0.5ex minus 1ex}
```

Space following title of a problem block:

```
\exerciseconfig{skipproblemstitle}{1.0ex plus 0ex minus 0.5ex}
```

Hook Code.

`insert...` Code to process data and to insert text at various points of processing.

Code to generate the title for a sheet; minimalistic default to display the sheet title:

```
\exerciseconfig{insertsheettitle}{\centerline{\getsheetdata{title}}}
```

Code to clear the page at the start and at the end of a new sheet:

```
\exerciseconfig{insertsheetclearpage}{\exercisecleardoublepage}
```

Code to insert before a sheet is displayed:

```
\exerciseconfig{insertsheetbefore}{}
```

Code to insert after a sheet is displayed:

```
\exerciseconfig{insertsheetafter}{}
```

Code to insert before a solution block is displayed:

```
\exerciseconfig{insertsolutionsbefore}{}
```

Code to insert after a solution block is displayed:

```
\exerciseconfig{insertsolutionsafter}{}
```

Code to insert before a problem block is displayed:

```
\exerciseconfig{insertproblemsbefore}{}
```

Code to insert after a problem block is displayed:

```
\exerciseconfig{insertproblemsafter}{}
```

Code to insert before a problem is displayed:

```
\exerciseconfig{insertproblembefore}{}
```

Code to insert after a problem is displayed:

```
\exerciseconfig{insertproblemafter}{}
```

Code to insert between a problem and its solution (if style `solutionbelow` is `problem`):

```
\exerciseconfig{insertproblemsolution}{}
```

Code to insert text into the problem opening line by means of `\addprobleminfo`:

```
\exerciseconfig{insertprobleminfo}{}
```

Code to determine the display of problems:

```
\exerciseconfig{insertproblemselect}[1]{}
```

Analogous definitions for subproblems:

```
\exerciseconfig{insertsubproblembefore}{}
\exerciseconfig{insertsubproblemafter}{}
\exerciseconfig{insertsubprobleminfo}{}
\exerciseconfig{insertsubproblemsolution}{}
\exerciseconfig{insertsubproblemselect}[1]{}
```

Analogous definitions for solutions:

```
\exerciseconfig{insertsolutionbefore}{}
\exerciseconfig{insertsolutionafter}{}
\exerciseconfig{insertsolutioninfo}{}
```

Text Composition for Environments.

`compose...` Macros to generate text for various situations. Preferably the output is plain text without formatting, but in some situations it may be required to address formatting in these macros.

Default separator for items:

```
\exerciseconfig{composeitemsep}{\ }
```

Compose sheet title; arguments are sheet number and raw title (empty if not specified); default is "Sheet `#1`" or given title "`#2`":

```
\exerciseconfig{composetitlesheet}[2]%
  {\exerciseifempty{#2}{\getexerciseconfig{termsheet} #1}{#2}}
```

Compose sheet title for pdf metadata; arguments are sheet number and raw title:

```
\exerciseconfig{composemetasheet}[2]%
  {\getexerciseconfig{composetitlesheet}{#1}{#2}}
```

Compose sheet title for table of contents; arguments are sheet number and raw title:

```
\exerciseconfig{composetocsheet}[2]%
  {\exerciseifempty{#2}{\getexerciseconfig{termsheet} #1}{#2}}
```

Problem item separator, delimiter:

```
\exerciseconfig{composeitemproblemsep}{\getexerciseconfig{composeitemsep}}
\exerciseconfig{composeitemproblemdelim}{.}
```

Compose problem item; argument is problem number:

```
\exerciseconfig{composeitemproblem}[1]%
  {#1\getexerciseconfig{composeitemproblemdelim}}
```

Unnamed problem separator, delimiter:

```
\exerciseconfig{composebareproblemsep}{ }
\exerciseconfig{composebareproblemdelim}{}
```

Compose unnamed problem title; arguments are problem number and name; includes exception for legacy behaviour:

```
\exerciseconfig{composebareproblem}[1]%
  {\getexerciseconfig{termproblem}\getexerciseconfig{composebareproblemsep}%
   \ifx\exf@config@composebareproblemdelim\exf@config@composeitemproblemdelim%
    \getexerciseconfig{composeitemproblem}{#1}\else%
    #1\getexerciseconfig{composebareproblemdelim}\fi}
```

Named problem separator, delimiter:

```
\exerciseconfig{composenamedproblemsep}{ }
\exerciseconfig{composenamedproblemdelim}{.}
```

Compose named problem title; arguments are problem number and name; includes exception for legacy behaviour:

```
\exerciseconfig{composenamedproblem}[2]%
  {\ifx\exf@config@composenamedproblemdelim\exf@config@composeitemproblemdelim%
     \getexerciseconfig{composeitemproblem}{#1}\else%
     #1\getexerciseconfig{composenamedproblemdelim}\fi%
   \getexerciseconfig{composenamedproblemsep}#2}
```

Compose problem title; arguments are problem number (empty if item is split off) and raw title (empty if not specified); default is “Problem `#1`” or “`#1. #2`”:

```
\exerciseconfig{composetitleproblem}[2]{\exerciseifempty{#1}%
   {\exerciseifempty{#2}{}{#2}}%
   {\exerciseifempty{#2}{\getexerciseconfig{composebareproblem}{#1}}%
    {\getexerciseconfig{composenamedproblem}{#1}{#2}}}}
```

Compose problem title for table of contents; arguments are problem number and raw title:

```
\exerciseconfig{composetocproblem}[2]%
  {\exerciseifempty{#2}{\getexerciseconfig{termproblem} #1}%
   {#1\getexerciseconfig{composenamedproblemdelim} #2}}
```

Subproblem item separator, delimiter:

```
\exerciseconfig{composeitemsubproblemsep}{\getexerciseconfig{composeitemsep}}
\exerciseconfig{composeitemsubproblemdelim}{}
```

Compose subproblem item; argument is subproblem number:

```
\exerciseconfig{composeitemsubproblem}[1]%
  {#1\getexerciseconfig{composeitemsubproblemdelim}}
```

Compose subproblem title; argument is subproblem number:

```
\exerciseconfig{composetitlesubproblem}[1]%
  {\getexerciseconfig{composeitemsubproblem}{#1}}
```

Solution item separator, delimiter:

```
\exerciseconfig{composeitemsolutionsep}{\getexerciseconfig{composeitemsep}}
\exerciseconfig{composeitemsolutiondelim}{:}
```

Compose solution item; arguments are problem and subproblem number:

```
\exerciseconfig{composeitemsolution}[2]{}
\exerciseconfig{composeitemsolutionsub}[2]%
  {\getexerciseconfig{composeitemsubproblem}{#2}}
```

Compose solution item for first solution in a problem section; arguments are problem and subproblem number:

```
\exerciseconfig{composeitemsolutionfirst}[2]%
  {\getexerciseconfig{composeitemsolution}{#1}{#2}}
\exerciseconfig{composeitemsolutionfirstsub}[2]%
  {\getexerciseconfig{composeitemsolutionsub}{#1}{#2}}
```

Compose title for single solution (solution intro if `solutionbelow` is `here` or `subproblem`); arguments are corresponding problem and subproblem number:

```
\exerciseconfig{composetitlesolutionsingle}[2]%
  {\getexerciseconfig{termsolution}%
   \getexerciseconfig{composeitemsolutiondelim}}
```

Compose title for one out of several solutions (solution intro if `solutionbelow` is `problem`, `sheet` or `manual`); arguments are corresponding problem and subproblem number:

```
\exerciseconfig{composetitlesolutionproblem}[1]{}
\exerciseconfig{composetitlesolutionsubproblem}[2]%
  {\getexerciseconfig{composeitemsubproblem}{#2}}
\exerciseconfig{composetitlesolutionmulti}[2]{\exerciseifempty{#2}%
  {\getexerciseconfig{composetitlesolutionproblem}{#1}}%
  {\getexerciseconfig{composetitlesolutionsubproblem}{#1}{#2}}}
```

Compose table of contents line for solution; arguments are problem number and raw title:

```
\exerciseconfig{composetocsolution}[2]%
  {\getexerciseconfig{composetocproblem}{#1}{#2}}
```

Compose title for solution block:

```
\exerciseconfig{composetitlesolutions}%
  {\getexerciseconfig{termsolutions}}
```

Compose title for problem block:

```
\exerciseconfig{composetitleproblems}%
  {\getexerciseconfig{termproblems}}
```

Compose table of contents line for solution block:

```
\exerciseconfig{composetocsolutions}%
  {\getexerciseconfig{composetitlesolutions}}
```

Compose table of contents line for problem block:

```
\exerciseconfig{composetocproblems}%
  {\getexerciseconfig{composetitleproblems}}
```

Compose sectional title for solution following a single problem (problem section intro if `solutionbelow` is `problem`); arguments are problem number and raw title:

```
\exerciseconfig{composetitlesolutionsproblemsingle}[2]%
  {\getexerciseconfig{termsolution}}
```

Compose sectional title for solution of one problem within a solution block (problem section intro if `solutionbelow` is `sheet` or `manual`); arguments are problem number and raw title:

```
\exerciseconfig{composetitlesolutionsproblemmulti}[2]%
  {\exerciseifempty{#2}{\getexerciseconfig{composebareproblem}{#1}}%
   {\getexerciseconfig{composenamedproblem}{#1}{#2}}}
```

Compose label:

```
\exerciseconfig{composeitemsolutionlabel}[2]{#1#2}
```

Points. Compose number of points:

```
\exerciseconfig{composepointsnum}[1]{#1}
```

Compose number of points followed by 'points'; use singular 'point' for 1:

```
\exerciseconfig{composepoints}[1]{\getexerciseconfig{composepointsnum}{#1}~%
  \ifdim #1pt=1pt\getexerciseconfig{termpoint}%
  \else\getexerciseconfig{termpoints}\fi}
```

Compose points declaration for use in opening line:

```
\exerciseconfig{composepointsstart}[1]{(\getexerciseconfig{composepoints}{#1})}
```

Compose points declaration for use in margin:

```
\exerciseconfig{composepointsmargin}[1]{\getexerciseconfig{composepoints}{#1}}
```

Compose points declaration for use in text:

```
\exerciseconfig{composepointsbody}[1]{(\getexerciseconfig{composepoints}{#1})}
```

Compose points declaration for use in sheet data:

```
\exerciseconfig{composepointssheet}[1]{%
  \exerciseifnotempty{#1}{\getexerciseconfig{composepoints}{#1}}}
```

Compose points declaration for solution with comment:

```
\exerciseconfig{composepointsaward}[2]%
  {(\getexerciseconfig{composepoints}{#1}\exerciseifnotempty{#2}{; #2})}
```

Compose alternative points for solution declaration with comment:

```
\exerciseconfig{composepointsawardalt}[2]%
  {(\getexerciseconfig{composepoints}{#1}*\exerciseifnotempty{#2}{; #2})}
```

Compose pairs of points; omit 0 components:

```
\exerciseconfig{composepointspair}[2]{%
  \ifdim#2pt=0pt%
   \getexerciseconfig{composepoints}{#1}%
  \else\ifdim#1pt=0pt%
   +\getexerciseconfig{composepoints}{#2}%
  \else%
   \getexerciseconfig{composepointsnum}{#1}+%
   \getexerciseconfig{composepointsnum}{#2}~%
   \getexerciseconfig{termpoints}%
  \fi\fi}
```

Compose pairs of points for designated use; recycle plain definition if no bonus points given:

```
\exerciseconfig{composepointspairbody}[2]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointsbody}{#1}\else%
  (\getexerciseconfig{composepointspair}{#1}{#2})\fi}
\exerciseconfig{composepointspairstart}[2]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointsstart}{#1}\else%
  (\getexerciseconfig{composepointspair}{#1}{#2})\fi}
\exerciseconfig{composepointspairmargin}[2]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointsmargin}{#1}\else%
  \getexerciseconfig{composepointspair}{#1}{#2}\fi}
\exerciseconfig{composepointspairsheet}[2]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointssheet}{#1}\else%
  \getexerciseconfig{composepointspair}{#1}{#2}\fi}
\exerciseconfig{composepointspairaward}[3]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointsaward}{#1}{#3}\else%
  (\getexerciseconfig{composepointspair}{#1}{#2}%
   \exerciseifnotempty{#3}{; #3})\fi}
\exerciseconfig{composepointspairawardalt}[3]{%
  \ifdim#2pt=0pt\getexerciseconfig{composepointsawardalt}{#1}{#3}\else%
  (\getexerciseconfig{composepointspair}{#1}{#2}*%
   \exerciseifnotempty{#3}{; #3})\fi}
```

Compose pairs of points for designated situations:

```
\exerciseconfig{composepointspairbodyproblem}[2]{%
  \getexerciseconfig{composepointspairbody}{#1}{#2}}
\exerciseconfig{composepointspairbodysubproblem}[2]{%
  \getexerciseconfig{composepointspairbody}{#1}{#2}}
\exerciseconfig{composepointspairbodysolution}[2]{%
  \getexerciseconfig{composepointspairbody}{#1}{#2}}
\exerciseconfig{composepointspairstartproblem}[2]{%
  \getexerciseconfig{composepointspairstart}{#1}{#2}}
\exerciseconfig{composepointspairstartsubproblem}[2]{%
  \getexerciseconfig{composepointspairstart}{#1}{#2}}
\exerciseconfig{composepointspairstartsolution}[2]{%
  \getexerciseconfig{composepointspairstart}{#1}{#2}}
```

Display points in the margin:

```
\exerciseconfig{insertpointsmargin}[1]{\marginpar{\footnotesize #1}}
```

Display warning about points mismatch:

```
\exerciseconfig{insertwarnpoints}[3]%
  {\textbf{points mismatch for #1 (#2 determined vs.\ #3 given)}}
```

Display warning about points changed:

```
\exerciseconfig{insertwarnpointsrerun}[1]%
  {\textbf{points changed for #1 (please recompile)}}
```

Counters. Define counter display via configuration interface:

```
\exerciseconfig{countersheet}{\arabic{\exf@sheetcounter}}
\exerciseconfig{counterproblem}{\arabic{\exf@problemcounter}}
\exerciseconfig{counterproblemmax}{10}
\exerciseconfig{countersubproblem}{\alph{\exf@subproblemcounter})}
\exerciseconfig{countersubproblemmax}{m)}
\exerciseconfig{countersheetequation}{\arabic{equation}}
\exerciseconfig{counterproblemequation}{P\arabic{equation}}
\exerciseconfig{countersolutionequation}{S\arabic{equation}}
```

Further Definitions.

`tagsheet` `tagproblem` `tagsubproblem` Templates for automatic generation of tags:

```
\exerciseconfig{tagsheet}{\arabic{\exf@sheetcounter}}
\exerciseconfig{tagproblem}{\csname the\exf@problemcounter\endcsname}
\exerciseconfig{tagsubproblem}{\problemtag-\arabic{\exf@subproblemcounter}}
```

`labelsheet` `labelproblem` `labelsubproblem` Templates for automatic generation of labels from tags:

```
\exerciseconfig{labelsheet}[1]{sheet:#1}
\exerciseconfig{labelproblem}[1]{prob:#1}
\exerciseconfig{labelsubproblem}[1]{\getexerciseconfig{labelproblem}{#1}}
```

`toclevel...` Table of contents levels for sheets, problems, solutions of problems and solution blocks; empty means no writing to table of contents:

```
\exerciseconfig{toclevelsheet}{}
\exerciseconfig{toclevelproblem}{}
\exerciseconfig{toclevelproblems}{}
\exerciseconfig{toclevelsolution}{}
\exerciseconfig{toclevelsolutions}{}
```

`extsolutions` `extproblems` Filename extension for solution and problem blocks:

```
\exerciseconfig{extsolutions}{.sol}
\exerciseconfig{extproblems}{.prb}
```

C.4 Styles

Styles are meant as a way to adjust several configuration options at the same time to achieve a consistent layout in some regard. Useful examples can be found among the extended exercise styles. They can serve a starting point for further custom styles.

Exercise Styles Code.

`\defexercisestylearg` Define a style with an argument:

```
\newcommand{\defexercisestylearg}[3][]{%
  \def\exf@tmp{#1}\ifx\exf@tmp\exf@empty%
   \define@key{exf@style}{#2}{#3}\else%
   \define@key{exf@style}{#2}[#1]{#3}\fi}
```

`\defexercisestyle` Define a style with a boolean argument; execute code onlf if true:

```
\newcommand{\defexercisestyle}[2]{%
  \exf@csdotwo\long\def{exf@style@code@#1}{#2}%
  \exf@exparg{\define@boolkey{exf@style}[exf@style@]{#1}[true]}%
   {\csname ifexf@style@#1\endcsname\csname exf@style@code@#1\endcsname\fi}}
```

`\exercisestyle` Process styles:

```
\newcommand{\exercisestyle}[1]{\exf@expsetkeys{exf@style}{#1}}
```

Default Exercise Styles.

`problemmanual` Delay display of problems:

```
\define@boolkey{exf@style}[exf@]{problemmanual}[true]{}
\exf@problemmanualfalse
```

`solutionbelow` Choose location for solutions:

```
\def\exf@solutionbelow{subproblem}
\define@choicekey{exf@style}{solutionbelow}%
  {here,subproblem,subproblem*,problem,problem*,sheet,manual}%
  {\ifexf@solfile@open\else\gdef\exf@solutionbelow{#1}\fi}
```

`sheetequation` `problemequation` `solutionequation` Use separate equation counters for sheets, problems and solutions:

```
\defexercisestyle{sheetequation}{}
\defexercisestyle{problemequation}{}
\defexercisestyle{solutionequation}{}
\exf@style@solutionequationtrue
```

`problempointsat` `subproblempointsat` `solutionpointsat` Choose where points of (sub)problems and solutions are displayed:

```
\def\exf@pointsat{start}
\define@choicekey{exf@style}{problempointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@pointsat{#1}}
\define@choicekey{exf@style}{pointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@pointsat{#1}}
\def\exf@subpointsat{end}
\define@choicekey{exf@style}{subproblempointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@subpointsat{#1}}
\define@choicekey{exf@style}{subpointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@subpointsat{#1}}
\def\exf@solpointsat{off}
\define@choicekey{exf@style}{solutionpointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@solpointsat{#1}}
\define@choicekey{exf@style}{solpointsat}%
  {start,start*,margin,end,manual,off}{\def\exf@solpointsat{#1}}
```

`problemby` `equationby` Declare problems or equations as subcounter of other counter:

```
\defexercisestylearg{problemby}{\exf@numberproblemwithin{#1}}
\defexercisestylearg{equationby}{\exf@numberequationwithin{#1}}
```

`pagebysheet` `problembysheet` `equationbysheet` Number pages, problems or equations by sheet:

```
\defexercisestyle{pagebysheet}{%
  \def\thepage{\csname the\exf@sheetcounter\endcsname.\arabic{page}}%
  \def\theHpage{\csname theH\exf@sheetcounter\endcsname.\arabic{page}}%
  \exerciseconfigappend{insertsheetbefore}{\setcounter{page}{1}}}
\defexercisestyle{problembysheet}%
  {\exf@numberproblemwithin{\exf@sheetcounter}}
\defexercisestyle{equationbysheet}%
  {\exf@numberequationwithin{\exf@sheetcounter}}
```

`fracpoints` Use vulgar fractions to display binary fractional points:

```
\defexercisestyle{fracpoints}%
  {\exerciseconfig{composepointsnum}[1]{\protect\showfracpoints{##1}}}
```

`twoside` Use two-sided layout for sheets:

```
\defexercisestylearg[true]{twoside}{\exercisesetup{twoside={#1}}}
```

`subproblemdelimitem` to append delimiter ')' to subproblem reference labels Use two-sided layout for sheets:

```
\defexercisestyle{subproblemdelimitem}%
  {\exerciseconfig{composeitemsubproblemdelim}{)}%
   \exerciseconfig{countersubproblem}{\alph{\exf@subproblemcounter}}%
   \exerciseconfig{countersubproblemmax}{m}}
```

Extended Exercise Styles. Declare more specific styles:

```
\ifexf@extstyle
```

`contents` Add sheets and problems to table of contents:

```
\defexercisestyle{contents}{%
  \exerciseconfig{toclevelsheet}{section}%
  \exerciseconfig{toclevelproblem}{subsection}}
```

`solutionsf` Use sans serif font for solutions:

```
\defexercisestyle{solutionsf}{%
  \exerciseconfigappend{styletextsolution}{\sffamily\let\itshape\slshape}}
```

`solutiondimproblem` Dim problem text if solutions are displayed:

```
\defexercisestyle{solutiondimproblem}{%
  \RequirePackage{color}%
  \exerciseconfigappend{styletextsolution}{\color[gray]{0}}%
  \exerciseconfigappend{styletextproblem}{\color[gray]{0.2}}}
```

`solutionsep` Separate solutions by horizontal lines:

```
\defexercisestyle{solutionsep}{%
  \exerciseconfig{insertsolutionsbefore}{\hrule\nopagebreak[3]\vspace{0.5ex}}%
  \exerciseconfig{insertsolutionsafter}%
   {\removelastskip\nopagebreak[3]\vspace{1.0ex}\hrule}}
```

`plainheader` Declare a simple sheet header with some configurable options; the configuration options `styleheader...` define font styles, `skipheaderbelow` the space below the header and `composeheaderbelow...` some auxiliary text to be displayed on the line below the header:

```
\defexercisestyle{plainheader}{%
  \exerciseconfig{styleheadertitle}{\Large\bfseries}%
  \exerciseconfig{styleheadercourse}{\sffamily}%
  \exerciseconfig{styleheaderbelow}{\footnotesize}%
  \exerciseconfig{skipheaderbelow}{3ex}%
  \exerciseconfig{composeheaderbelowleft}{}%
  \exerciseconfig{composeheaderbelowright}{}%
  \exerciseconfig{composeheaderbelowcenter}{}%
  \exerciseconfig{insertsheettitle}{\noindent%
   \begin{minipage}{\textwidth}%
   {\getexerciseconfig{styleheadertitle}%
    \makebox[0pt][l]{\getexercisedata{course}}%
    \hfill\makebox[0pt][r]{\getsheetdata{title}}\par}%
   {\getexerciseconfig{styleheadercourse}%
    \makebox[0pt][l]{\getexercisedata{institution}%
```

```
    \exercisedataempty{period}{}{, \getexercisedata{period}}}%
   \hfill\makebox[0pt][r]{\getexercisedata{instructor}}%
   \vphantom{g}\par}%
  \hrule%
  {\def\tmp{}%
   \exerciseconfigempty{composeheaderbelowleft}{}{\def\tmp{.}}%
   \exerciseconfigempty{composeheaderbelowcenter}{}{\def\tmp{.}}%
   \exerciseconfigempty{composeheaderbelowright}{}{\def\tmp{.}}%
   \exerciseifnotempty{\tmp}%
    {\getexerciseconfig{styleheaderbelow}\vphantom{\^A}%
     \makebox[0pt][l]{\getexerciseconfig{composeheaderbelowleft}}%
     \hfill\makebox[0pt][c]{\getexerciseconfig{composeheaderbelowcenter}}%
     \hfill\makebox[0pt][r]{\getexerciseconfig{composeheaderbelowright}}%
     \vspace*{-\baselineskip}\vspace*{-\parskip}\par}}%
  \end{minipage}%
  \par\addvspace{\getexerciseconfig{skipheaderbelow}}}}
```

Done with extended styles:

```
\fi
```

C.5 Metadata

Global Metadata Code.

`\defexercisedata` Declare global metadata field by defining a key *key* in category `exf@data` that stores the chosen value in `\exf@data@`*key*:

```
\newcommand{\defexercisedata}[1]{%
  \exf@csdo\def{exf@data@#1}{}%
  \define@key{exf@data}{#1}%
   {\exf@csdo\gdef{exf@data@#1}{##1}}}
```

`\exercisedata` Process key-value pairs:

```
\newcommand{\exercisedata}[1]{\setkeys{exf@data}{#1}}
```

`\getexercisedata` Read global metadata:

```
\newcommand{\getexercisedata}[1]{\csname exf@data@#1\endcsname}
```

`\exercisedataempty` Check whether the field is empty:

```
\newcommand{\exercisedataempty}[3]{\exf@csdo\ifx{exf@data@#1}\exf@empty%
  #2\else#3\fi}
```

Global Metadata Declarations. Declare fields corresponding to standard pdf metadata:

```
\defexercisedata{author}
\defexercisedata{title}
\defexercisedata{subject}
\defexercisedata{keyword}
```

Declare additional general purpose fields:

```
\defexercisedata{date}
```

Declare metadata related to courses:

```
\defexercisedata{instructor}
\defexercisedata{course}
\defexercisedata{institution}
\defexercisedata{period}
\defexercisedata{material}
```

Overwrite standard definitions for `author`, `title`, `date` to also fill ordinary LaTeX structures:

```
\define@key{exf@data}{author}{\gdef\exf@data@author{#1}\author{#1}}
\define@key{exf@data}{title}{\gdef\exf@data@title{#1}\title{#1}}
\define@key{exf@data}{date}{\gdef\exf@data@date{#1}\date{#1}}
```

Sheet Metadata.

`\defsheetdata` Declare sheet metadata field by defining a key *key* in category `exf@sheet` that stores the chosen value in `\exf@data@sheet@`*key*:

```
\newcommand{\defsheetdata}[1]{%
  \exf@csdo\def{exf@data@sheet@#1}{}%
  \define@key{exf@sheet}{#1}%
   {\exf@csdo\def{exf@data@sheet@#1}{##1}}}
```

`\setsheetdata` Set sheet metadata:

```
\newcommand{\setsheetdata}[1]{\setkeys{exf@sheet}{#1}}
```

`\getsheetdata` Read sheet metadata:

```
\newcommand{\getsheetdata}[1]{\csname exf@data@sheet@#1\endcsname}
```

`\sheetdataempty` Check whether the field is empty:

```
\newcommand{\sheetdataempty}[3]{\exf@csdo\ifx{exf@data@sheet@#1}\exf@empty%
  #2\else#3\fi}
```

Declare general purpose fields:

```
\defsheetdata{due}
\defsheetdata{handout}
\defsheetdata{editdate}
\defsheetdata{author}
\defsheetdata{editor}
```

Special title processing:

```
\def\exf@data@sheet@rawtitle{}
\define@key{exf@sheet}{title}{\def\exf@data@sheet@rawtitle{#1}}
\def\exf@data@sheet@title{\exf@config@composetitlesheet%
  {\csname the\exf@sheetcounter\endcsname}{\exf@data@sheet@rawtitle}}%
```

Special points processing:

```
\def\exf@data@sheet@points{\ifdefined\exf@sheet@points%
   \expandafter\exf@config@composepointspairsheet\exf@sheet@points\fi}%
```

Problem Metadata.

`\defproblemdata` Declare problem metadata field by defining a key *key* in category `exf@problem` that stores the chosen value in `\exf@data@problem@`*key*:

```
\newcommand{\defproblemdata}[1]{%
  \exf@csdo\def{exf@data@problem@#1}{}%
  \define@key{exf@problem}{#1}%
   {\exf@csdo\def{exf@data@problem@#1}{##1}}}
```

`\setproblemdata` Set problem metadata:

```
\newcommand{\setproblemdata}[1]{\setkeys{exf@problem,exf@scanproblem}{#1}}
```

`\setsubproblemdata` Set subproblem metadata:

```
\newcommand{\setsubproblemdata}[1]{%
  \setkeys{exf@subproblem,exf@scansubproblem}{#1}}
```

`\getproblemdata` Read problem metadata:

```
\newcommand{\getproblemdata}[1]{\csname exf@data@problem@#1\endcsname}
```

`\problemdataempty` Check whether the field is empty:

```
\newcommand{\problemdataempty}[3]{\exf@csdo\ifx{exf@data@problem@#1}\exf@empty%
  #2\else#3\fi}
```

Special title processing:

```
\def\exf@data@problem@rawtitle{}
\define@key{exf@problem}{title}{\def\exf@data@problem@rawtitle{#1}}
\def\exf@data@problem@title{\exf@config@composetitleproblem{%
 \csname the\exf@problemcounter\endcsname}{\exf@data@problem@rawtitle}}%
```

Problem Environment Code.

`\exf@addmargin` Define a length for environment margin:

```
\newlength\exf@addmargin
```

`\exf@section` Write out problem opening line followed by some amount of skip (positive dimensions add vertical space, negative dimensions add horizontal space); protected expand argument if in horizontal mode (because it will be held until text is output and some definitions may become invalid):

```
\newcommand{\exf@section}[2]{\setlength\exf@tmpdim{#1}%
  \ifdim\exf@tmpdim<0pt%
   \protected@edef\exf@tmp{#2}%
  \else%
   \def\exf@tmp{#2}%
  \fi%
  \exf@exparg{\@startsection{}{}{0pt}{0pt}{#1}{}*}{\exf@tmp}}
```

`\exf@init@block` Clean info buffer, define amount of skip between items:

```
\newcommand{\exf@init@block}[1]{%
  \def\exf@intro{}\def\exf@intro@skip{#1}%
  \exf@addmargin0pt\def\exf@introitem{}}
```

`\exf@append@intro` Append to info buffer:

```
\newcommand{\exf@append@intro}[1]%
  {\exf@append@def\exf@intro{#1\hspace{\exf@intro@skip}}}
```

`\exf@prepend@intro` Prepend to info buffer:

```
\newcommand{\exf@prepend@intro}[1]%
  {\exf@prepend@def\exf@intro{#1\hspace{\exf@intro@skip}}}
```

`\exf@open@block` Open environment, set margin, compose opening line:

```
\newcommand{\exf@open@block}[1]{%
  \advance\leftskip\exf@addmargin%
  \advance\linewidth-\exf@addmargin%
  \advance\@totalleftmargin\exf@addmargin%
  \ifx\exf@intro\exf@empty%
   \exf@section{0pt}{\exf@introitem}%
  \else%
   \exf@section{#1}{\exf@introitem\exf@intro\unskip}%
  \fi}%
```

`\exf@close@block` Close environment, undo margin:

```
\newcommand{\exf@close@block}{%
  \advance\leftskip-\exf@addmargin%
  \advance\linewidth\exf@addmargin%
  \advance\@totalleftmargin-\exf@addmargin}%
```

`\addprobleminfo` Interface to append or prepend to info buffer:

```
\newcommand{\addprobleminfo}{\@ifstar\exf@prepend@intro\exf@append@intro}
```

`\exf@addinfoswitch` Add a switch for displaying problem info:

```
\newcommand{\exf@addinfoswitch}[1]%
  {\define@boolkey{exf@infoswitch}[exf@showdata@]{#1}[true]{}}
```

`\defprobleminfo` Declare a problem info field, add corresponding info switch, process key-value pair if switch activated:

```
\newcommand{\defprobleminfo}[2]{%
  \exf@addinfoswitch{#1}%
  \exerciseconfig{compose@probleminfo@#1}[1]{#2}%
  \exf@exparg{\define@key{exf@probleminfo}{#1}}%
   {\csname ifexf@showdata@#1\endcsname\exf@append@intro{%
    \csname exf@config@compose@probleminfo@#1\endcsname{##1}}\fi}}
```

`\showprobleminfo` Process info switches, expand argument first:

```
\newcommand{\showprobleminfo}[1]{\exf@expsetkeys{exf@infoswitch}{#1}}
```

Problem Info Declarations. Declare general purpose fields:

```
\defprobleminfo{optional}{\emph{#1:}}
\showprobleminfo{optional}
\defprobleminfo{difficulty}{(#1)}
```

Declare fields for internal information (mostly):

```
\defprobleminfo{comment}{#1}
\defprobleminfo{author}{$\langle$#1$\rangle$}
\defprobleminfo{editor}{$\{$#1$\}$}
\defprobleminfo{source}{[#1]}
\defprobleminfo{keyword}{\#(#1)}
```

Declare more specific fields:

```
\ifexf@extdata
\defprobleminfo{review}{#1}
\defprobleminfo{recycle}{[[#1]]}
\defprobleminfo{timesolve}{\{#1\}}
\defprobleminfo{timepresent}{\{\!\{#1\}\!\}}
\fi
```

Write Metadata to PDF Files.

`\exf@writemetadata` Write Metadata to PDF Files in case hyperref is available:

```
\newcommand{\exf@writemetadata}{%
  \ifdefined\hypersetup%
```

Write author, title, subject and keywords:

```
   \ifx\exf@data@author\exf@empty\else%
    \hypersetup{pdfauthor={\exf@data@author}}\fi%
   \ifx\exf@data@title\exf@empty\else%
    \hypersetup{pdftitle={\exf@data@title}}\fi%
   \ifx\exf@data@subject\exf@empty\else%
    \hypersetup{pdfsubject={\exf@data@subject}}\fi%
   \ifx\exf@data@keyword\exf@empty\else%
    \hypersetup{pdfkeywords={\exf@data@keyword}}\fi%
  \fi}
```

Automatic writing at `\begin{document}`:

```
\AtBeginDocument{\exf@ifis\exf@metadata{auto}%
  {\exf@writemetadata\gdef\exf@metadata{off}}}
```

`\writeexercisedata` Write metadata manually:

```
\newcommand{\writeexercisedata}{\exf@ifis\exf@metadata{manual}%
  {\exf@writemetadata\gdef\exf@metadata{off}}}
```

C.6 Counters

`sheet` `problem` `subproblem` `solution` Define main counters (with customised names if necessary) and equation counters:

```
\newcounter{\exf@sheetcounter}
\newcounter{\exf@problemcounter}
\newcounter{\exf@subproblemcounter}[\exf@problemcounter]
\newcounter{\exf@solutioncounter}[\exf@problemcounter]
\newcount\exf@eqsav
\newcounter{exf@sheetequation}
\newcounter{exf@problemequation}
\newcounter{exf@solutionequation}
```

Implement counter display; take care of corresponding hyperref labels:

```
\exf@csdo\def{the\exf@sheetcounter}{\exf@config@countersheet}
\exf@csdo\def{the\exf@problemcounter}{\exf@config@counterproblem}
\exf@csdo\def{the\exf@subproblemcounter}{\exf@config@countersubproblem}
\def\theexf@sheetequation{\exf@config@countersheetequation}
\def\theHexf@sheetequation{sheet.\arabic{equation}}
\def\theexf@problemequation{\exf@config@counterproblemequation}
\def\theHexf@problemequation{prob.\arabic{equation}}
```

```
\def\theexf@solutionequation{\exf@config@countersolutionequation}
\def\theHexf@solutionequation{sol.\arabic{equation}}
```

`@numberproblemwithin` Declare problem counter as subcounter of `#1`:

```
\newcommand{\exf@numberproblemwithin}[1]{%
  \@addtoreset{\exf@problemcounter}{#1}%
  \exf@csdo\def{the\exf@problemcounter}%
   {\csname the#1\endcsname.\exf@config@counterproblem}%
  \edef\exf@tmp{#1}%
  \ifx\exf@tmp\exf@sheetcounter%
   \exerciseconfig{tagproblem}{\ifdefined\sheettag\sheettag-\fi%
    \arabic{\exf@problemcounter}}%
  \else%
   \exerciseconfig{tagproblem}{\csname the#1\endcsname-%
    \arabic{\exf@problemcounter}}%
  \fi}
```

`numberequationwithin` Declare various new equation counters as subcounter of `#1`; take care of corresponding hyperref labels:

```
\newcommand{\exf@numberequationwithin}[1]{%
  \@addtoreset{exf@sheetequation}{#1}%
  \def\theexf@sheetequation%
   {\csname the#1\endcsname.\exf@config@countersheetequation}%
  \def\theHexf@sheetequation%
   {\csname theH#1\endcsname.sheet.\arabic{equation}}%
  \@addtoreset{exf@problemequation}{#1}%
  \def\theexf@problemequation%
   {\csname the#1\endcsname.\exf@config@counterproblemequation}%
  \def\theHexf@problemequation%
   {\csname theH#1\endcsname.prob.\arabic{equation}}%
  \@addtoreset{exf@solutionequation}{#1}%
  \def\theexf@solutionequation%
   {\csname the#1\endcsname.\exf@config@countersolutionequation}%
  \def\theHexf@solutionequation%
   {\csname theH#1\endcsname.sol.\arabic{equation}}}
```

C.7 Buffers

File Output.

`\ifexf@solfile@open` Conditional whether output files are presently in use:

```
\newif\ifexf@solfile@open\exf@solfile@openfalse
\newif\ifexf@probfile@open\exf@probfile@openfalse
```

`\exf@solfile` Reserve file handles:

```
\newwrite\exf@solfile
\newwrite\exf@probfile
```

`\exf@writeline` Write a line to the file:

```
\newcommand{\exf@writeline}[2]{\immediate\write#1{#2}}
```

`\exf@linesep` Return a separator line:

```
\newcommand{\exf@linesep}%
  {\@percentchar----------------------------------------}
```

`\exf@lineno` Return current position in source file; display line number and source file name (if available via package currfile):

```
\newcommand{\exf@lineno}{\@percentchar%
  \ifdefined\currfilename\currfilename\space\fi%
  l.\the\inputlineno}
```

`\exf@start@solfile` Open a new solution file `#1.sol` (do nothing if already open); indicate source, switch to manual solution display mode:

```
\newcommand{\exf@start@solfile}[1]{%
  \ifexf@solfile@open\else%
   \exercisestyle{solutionbelow=manual}%
   \global\exf@solfile@opentrue%
   \edef\exf@tmp{#1}%
   \immediate\openout\exf@solfile\exf@tmp\exf@config@extsolutions\relax%
   \exf@writeline\exf@solfile{\@percentchar%
    generated from file `\jobname' by exframe.sty}%
   \ifexf@lineno\exf@writeline\exf@solfile{\exf@lineno}\fi%
   \exf@writeline\exf@solfile{}%
  \fi}
```

`\exf@close@solfile` Close solution file (if open); indicate position, close and reset variables:

```
\newcommand{\exf@close@solfile}{%
  \ifexf@solfile@open%
   \ifexf@lineno\exf@writeline\exf@solfile{\exf@linesep}%
    \exf@writeline\exf@solfile{\exf@lineno}\fi%
   \exf@writeline\exf@solfile{\@backslashchar endinput}%
   \immediate\closeout\exf@solfile%
   \global\exf@solfile@openfalse%
  \fi}
```

`\exf@start@probfile` Open a new problem file `#1.prb` (do nothing if already open); indicate source, switch to manual problem display mode:

```
\newcommand{\exf@start@probfile}[1]{%
  \ifexf@probfile@open\else%
   \global\exf@probfile@opentrue%
   \edef\exf@tmp{#1}%
   \immediate\openout\exf@probfile\exf@tmp\exf@config@extproblems\relax%
   \exf@writeline\exf@probfile{\@percentchar%
    generated from file `\jobname' by exframe.sty}%
   \ifexf@lineno\exf@writeline\exf@probfile{\exf@lineno}\fi%
   \exf@writeline\exf@probfile{}%
  \fi}
```

`\exf@close@probfile` Close problem file (if open); indicate position, close and reset variables:

```
\newcommand{\exf@close@probfile}{%
  \ifexf@probfile@open%
   \ifexf@lineno\exf@writeline\exf@probfile{\exf@linesep}%
    \exf@writeline\exf@probfile{\exf@lineno}\fi%
   \exf@writeline\exf@probfile{\@backslashchar endinput}%
   \immediate\closeout\exf@probfile%
   \global\exf@probfile@openfalse%
  \fi}
```

Make sure to properly close files at the end:

```
\AtEndDocument{\exf@close@solfile\exf@close@probfile}
```

Buffers.

\exf@solbuf
\exf@probbuf
\exf@subprobbuf
\ifexf@solbuf@clean
\ifexf@probbuf@clean

Declare token buffers for storing problems and solutions and conditionals indicating whether the buffers have been used:

```
\newtoks\exf@solbuf
\newtoks\exf@probbuf
\newtoks\exf@subprobbuf
\newif\ifexf@solbuf@clean\exf@solbuf@cleantrue
\newif\ifexf@probbuf@clean\exf@probbuf@cleantrue
```

\exf@clear@solbuf
\exf@clear@probbuf
exf@clear@subprobbuf

Clear a buffer and mark clean:

```
\def\exf@clear@solbuf{\global\exf@solbuf@cleantrue\global\exf@solbuf={}}
\def\exf@clear@probbuf{\global\exf@probbuf@cleantrue\global\exf@probbuf={}}
\def\exf@clear@subprobbuf{\global\exf@subprobbuf={}}
```

\exf@append@buf

Append tokens to buffer:

```
\def\exf@append@buf#1#2{\global#1=\expandafter{\the#1#2}}
```

\exf@addline

Add a protected expanded line to the buffer:

```
\def\exf@addline#1#2{{\protected@edef\exf@tmp{#2}%
  \exf@exparg{\exf@append@buf#1}{\exf@tmp^^J}}}
```

\exf@source@buf

Source a buffer into the document:

```
\def\exf@source@buf#1{\exf@exptwo\scantokens{\the#1}}
```

\exf@write@buf

Write the buffer into the solution file:

```
\def\exf@write@buf#1#2{\exf@writeline#1{\the#2}}
```

Verbatim Processing.

\exf@verbatim

Start reading the buffer from the environment body:

```
\newcommand{\exf@verbatim}{%
  \begingroup%
  \@bsphack%
  \let\do\@makeother\dospecials%
  \catcode`\^^M\active%
  \def\verbatim@processline{\exf@exptwo\exf@verbatim@process%
   {\the\verbatim@line^^J}}%
  \verbatim@start}
```

\exf@endverbatim

Stop reading the buffer:

```
\newcommand{\exf@endverbatim}{\@esphack\endgroup}
```

\exf@scanblock

Scan an optional argument from a verbatim environment; allow for an empty environment and an empty first line; argument #1 is macro to be called eventually:

```
\def\exf@scanblock#1{%
```

Check for empty first line:

```
\@ifnextchar\par{\exf@scanblock@par{#1}}{\exf@scanblock@sel{#1}}}
```

Handle empty first line, implies no optional argument:

```
\long\def\exf@scanblock@par#1\par{\exf@scanblock@sel{#1}[]}
```

Check for optional argument ([) and for environment ending (\end):

```
\def\exf@scanblock@sel#1{\@ifnextchar[{\exf@scanblock@opt{#1}}%
  {\@ifnextchar\end{\exf@scanblock@end{#1}}{\exf@scanblock@noopt{#1}}}}
```

Handle empty environment, hopefully environment matches (otherwise?!):

```
\def\exf@scanblock@end#1\end#2{%
  \def\exf@tmp{#2}\ifx\exf@tmp\@currenvir%
    \def\exf@verbatim{}\def\exf@endverbatim{}%
  \fi%
  #1{}{\scantokens{\end{#2}}}}
```

Pass on without and with optional argument; pass on optional argument and any token scanned prematurely:

```
\def\exf@scanblock@noopt#1#2{#1{}{\scantokens#2}}
\def\exf@scanblock@opt#1[#2]{#1{#2}{}}
```

C.8 Points

Points Arithmetic.

\exf@addtopoints Add points #2+#3 to macro #1 using metric register:

```
\newcommand{\exf@addtopoints}[3]{%
  \ifdefined#1\else\def#1{{0}{0}}\fi%
  \setlength\exf@tmpdim{\expandafter\@firstoftwo#1pt}%
  \addtolength\exf@tmpdim{#2pt}%
  \edef\exf@tmp{\strip@pt\exf@tmpdim}%
  \setlength\exf@tmpdim{\expandafter\@secondoftwo#1pt}%
  \addtolength\exf@tmpdim{#3pt}%
  \xdef#1{{\exf@tmp}{\strip@pt\exf@tmpdim}}}
```

\exf@pointsmismatch Execute #3 if points disagree:

```
\newcommand{\exf@pointsmismatch}[3]{%
  \let\exf@tmp\@undefined%
  \ifdim\expandafter\@firstoftwo#1pt=\expandafter\@firstoftwo#2pt\else%
   \def\exf@tmp{}\fi%
  \ifdim\expandafter\@secondoftwo#1pt=\expandafter\@secondoftwo#2pt\else%
   \def\exf@tmp{}\fi%
  \ifdefined\exf@tmp#3\fi}
```

Points Expansion.

\exf@outpoints If points macro #3 is set, expand #3, pass on as #1{\protect#2}#3 and clear #3 globally:

```
\def\exf@outpoints#1#2#3{\ifdefined#3%
   \exf@exptwo\exf@outpoints@switch{#3}{#1}{#2}%
   \global\let#3\@undefined\fi}
\def\exf@outpoints@switch#1#2#3{#2{\protect#3#1}}
```

`\exf@scanpoints` Call as `\exf@scanpoints#1[`*regular*`][+`*bonus*`]++@` to write `{`*regular*`}{`*bonus*`}` into `#1`; fill with 0 if empty:

```
\def\exf@scanpoints#1#2+#3+#4@{%
  \edef#1{\if @#2@0\else#2\fi}%
  \edef#1{{#1}{\if @#3@0\else#3\fi}}}
```

`\exf@formatpoints` Format as [`#1`][`+#2`]; remove 0 components:

```
\def\exf@formatpoints#1#2{\ifdim#2pt=0pt#1\else%
  \ifdim#1pt=0pt+#2\else#1+#2\fi\fi}
```

`\extractpoints` Extract main (plain) or bonus (starred) part from saved points register:

```
\newcommand{\extractpoints}{\@ifstar{\exf@extractpoints\@secondoftwo}%
  {\exf@extractpoints\@firstoftwo}}
\newcommand{\exf@extractpoints}[2]{\edef\exf@tmp{#2}%
  \exf@exptwo\exf@scanpoints\exf@tmp\exf@tmp++@%
  \expandafter#1\exf@tmp}
```

`\switchpoints` Extract main (plain) and bonus (starred) part from points, and execute one of three:

```
\newcommand{\switchpoints}[5]{\edef\exf@tmp{#5}%
  \exf@exptwo\exf@scanpoints\exf@tmp\exf@tmp++@%
  \expandafter\exf@switchpoints\exf@tmp{#1}{#2}{#3}{#4}}
\newcommand{\exf@switchpoints}[6]{%
  \ifdim#2pt=0pt\ifdim#1pt=0pt\def\exf@tmp##1##2{#6}%
   \else\def\exf@tmp##1##2{#3}\fi%
  \else\ifdim#1pt=0pt\def\exf@tmp##1##2{#4}%
   \else\def\exf@tmp##1##2{#5}\fi\fi\exf@tmp{#1}{#2}}
```

Tools.

`exf@makepointsmargin` Combination to typeset points in margin:

```
\newcommand{\exf@makepointsmargin}[2]{%
  \exf@config@insertpointsmargin{\exf@config@composepointspairmargin{#1}{#2}}}
```

`\exf@warnmismatch` If points `#3` and `#4` are defined and disagree, issue a warning message:

```
\newcommand{\exf@warnmismatch}[4]{%
  \ifdefined#4\ifdefined#3\exf@pointsmismatch#3#4{%
   \let\exf@tmp\PackageWarning%
   \ifx#1\exf@solutionname\let\exf@tmp\PackageWarningNoLine\fi%
   \exf@tmp{exframe}{points mismatch %
    (\expandafter\exf@formatpoints#3 determined %
    vs. \expandafter\exf@formatpoints#4 given) %
    for #1 \csname the#2\endcsname}%
   \ifexf@warntext\edef\exf@tmp{%
    {\expandafter\exf@formatpoints#3}{\expandafter\exf@formatpoints#4}}%
    \exf@exptwo\exf@config@insertwarnpoints#1\exf@tmp\fi}%
  \fi\fi}
```

`\exf@warnrerun` If points `#3` and `#4` are defined and disagree, issue advice to recompile:

```
\newcommand{\exf@warnrerun}[4]{%
  \ifdefined#4\ifdefined#3\exf@pointsmismatch#3#4{%
   \PackageWarning{exframe}{points changed %
    for #1 \csname the#2\endcsname; rerun to fix}%
```

```
    \ifexf@warntext\exf@config@insertwarnpointsrerun#1\fi}%
  \fi\fi}
```

Binary Rational Numbers.

`\exf@splitsign` `\exf@splitdecimal` Split a decimal float number into sign, integer and fractional part:

```
\def\exf@splitsign#1-#2-#3&{\def\exf@splitnum{#1#2}\def\exf@splitminus{#3}}
\def\exf@splitdecimal#1.#2.#3&{\def\exf@splitint{#1}\def\exf@splitdec{#2}}
```

`\showfracpoints` Display a float number as a fraction with denominators 2, 4 or 8 when possible; first split number, complete missing zeros and handle cases:

```
\newcommand{\showfracpoints}[1]{%
  \edef\exf@tmp{#1}%
  \expandafter\exf@splitsign\exf@tmp--&%
  \expandafter\exf@splitdecimal\exf@splitnum..&%
  \if @\exf@splitint @\def\exf@splitint{0}\fi%
  \if @\exf@splitdec @\def\exf@splitdec{0}\fi%
  \def\exf@tmp{\exf@splitint.\exf@splitdec}%
  \ifnum\exf@splitdec=0\def\exf@tmp{\exf@splitint}\fi%
  \ifnum\exf@splitdec=5\def\exf@tmp{\exf@config@frac{\exf@splitint}{1}{2}}\fi%
  \ifnum\exf@splitdec=25\def\exf@tmp{\exf@config@frac{\exf@splitint}{1}{4}}\fi%
  \ifnum\exf@splitdec=75\def\exf@tmp{\exf@config@frac{\exf@splitint}{3}{4}}\fi%
  \ifnum\exf@splitdec=125\def\exf@tmp{\exf@config@frac{\exf@splitint}{1}{8}}\fi%
  \ifnum\exf@splitdec=375\def\exf@tmp{\exf@config@frac{\exf@splitint}{3}{8}}\fi%
  \ifnum\exf@splitdec=625\def\exf@tmp{\exf@config@frac{\exf@splitint}{5}{8}}\fi%
  \ifnum\exf@splitdec=875\def\exf@tmp{\exf@config@frac{\exf@splitint}{7}{8}}\fi%
  \ifx\exf@splitminus\exf@empty\else$\exf@splitminus$\fi\exf@tmp}
```

`\exf@config@frac` Display a vulgar fraction such as $12^3/_4$:

```
\newcommand{\exf@config@frac}[3]{%
  \ifnum#1=0\relax\else#1\fi%
  \ifnum#2=0\relax\else$%
   ^{\exf@text{#2}}%
   \mskip-4mu/\mskip-2mu%
   _{\exf@text{#3}}$\fi}
```

Sheet Points Code.

`notedata@sheetpoints` Store a sheet point number in a macro:

```
\newcommand{\exf@notedata@sheetpoints}[2]{%
  \exf@csdo\gdef{exf@sheetpoints@#1}{#2}}
```

`exf@writesheetpoints` Write sheet points to the `.aux` file:

```
\newcommand{\exf@writesheetpoints}[2]%
  {\exf@writedata{sheetpoints}{\sheettag}{\exf@formatpoints{#1}{#2}}}
```

`\getsheetpoints` Read points for current sheet or from `.aux` file:

```
\newcommand{\getsheetpoints}[1]{\if @#1@%
   \ifdefined\exf@sheet@points%
    \expandafter\exf@formatpoints\exf@sheet@points\else 0\fi%
  \else\exf@csor{exf@sheetpoints@#1}{0}\fi}
```

Problem Points Code.

tedata@problempoints Store a problem point number in a macro:

```
\newcommand{\exf@notedata@problempoints}[2]{%
  \exf@csdo\gdef{exf@problempoints@#1}{#2}}
```

f@writeproblempoints Write problem points to the .aux file:

```
\newcommand{\exf@writeproblempoints}[2]%
  {\exf@writedata{problempoints}{\problemtag}{\exf@formatpoints{#1}{#2}}}
```

\getproblempoints Read points for current problem or from .aux file:

```
\newcommand{\getproblempoints}[1]{\if @#1@%
  \ifdefined\exf@problem@points%
   \expandafter\exf@formatpoints\exf@problem@points\else 0\fi%
  \else\exf@csor{exf@problempoints@#1}{0}\fi}
```

\showpoints Show points within a problem or subproblem:

```
\newcommand{\showpoints}{%
  \ifdefined\exf@in@solution\exf@outpoints{\exf@ensuretext}%
   {\exf@config@composepointspairbodysolution}{\exf@solution@points@show}%
  \else\ifdefined\exf@in@subproblem\exf@outpoints{\exf@ensuretext}%
   {\exf@config@composepointspairbodysubproblem}{\exf@subproblem@points@show}%
  \else\ifdefined\exf@in@problem\exf@outpoints{\exf@ensuretext}%
   {\exf@config@composepointspairbodyproblem}{\exf@problem@points@show}%
  \fi\fi\fi}
```

Subproblem Points Code.

ata@subproblempoints Store a subproblem point number in a macro:

```
\newcommand{\exf@notedata@subproblempoints}[2]{%
  \exf@csdo\gdef{exf@subproblempoints@#1}{#2}}
```

ritesubproblempoints Write subproblem points to the .aux file:

```
\newcommand{\exf@writesubproblempoints}[2]%
  {\exf@writedata{subproblempoints}%
   {\subproblemtag}{\exf@formatpoints{#1}{#2}}}
```

\getsubproblempoints Read points for current subproblem:

```
\newcommand{\getsubproblempoints}[1]{\if @#1@%
  \ifdefined\exf@subproblem@points%
   \expandafter\exf@formatpoints\exf@subproblem@points\else 0\fi%
  \else\exf@csor{exf@subproblempoints@#1}{0}\fi}
```

Solution Points Code.

\exf@awardpointsalt Award points for alternative or optional solution; does not count towards solution total:

```
\newcommand{\exf@awardpointsalt}[2][]{\exf@scanpoints\exf@tmp#2++@%
  \exf@exptwo\exf@ensuretext{%
   \expandafter\exf@config@composepointspairawardalt\exf@tmp{#1}}}
```

\exf@awardpointsreg Award points for regular solution; counts towards solution total:

```
\newcommand{\exf@awardpointsreg}[2][]{\exf@scanpoints\exf@tmp#2++@%
  \exf@exptwo\exf@addtopoints\exf@solution@points@total\exf@tmp%
  \exf@scanpoints\exf@tmp#2++@%
  \exf@exptwo\exf@ensuretext{%
   \expandafter\exf@config@composepointspairaward\exf@tmp{#1}}}
```

\awardpoints Award points within solution with optional starred form:

```
\newcommand{\awardpoints}{\@ifstar\exf@awardpointsalt\exf@awardpointsreg}
```

\getsolutionpoints Read points for current solution:

```
\newcommand{\getsolutionpoints}[1]{\if @#1@%
   \ifdefined\exf@solution@points%
    \expandafter\exf@formatpoints\exf@solution@points\else 0\fi%
  \else 0\fi}
```

C.9 Tag Lists

Loop Lists.

exerciseloop
exf@loopdepth Counters for item number within list and depth of loop nesting:

```
\newcounter{exerciseloop}
\newcounter{exf@loopdepth}
```

exf@listwalk Step through a list; call the callback function #1 with current item #2:

```
\def\exf@listwalk#1#2{\if @#2@\def\exf@tmp{}\else%
  \def\exf@tmp{#1{#2}\exf@listwalk#1}\fi\exf@tmp}
```

\exerciseloop Loop through a list of items in braces #1 which is expanded first; store code #2 to be executed in a callback function (need to use different callback function for each loop depth, callback function must be global to work within table cells); initialise item counter:

```
\newcommand{\exerciseloop}[2]{\addtocounter{exf@loopdepth}{1}%
  \setcounter{exerciseloop}{0}%
  \exf@csdo\gdef{exf@listcallback@\roman{exf@loopdepth}}##1%
   {\stepcounter{exerciseloop}#2}%
  \edef\exf@tmp{#1}%
  \exf@csdotwo\exf@exptwo\exf@listwalk%
   {exf@listcallback@\roman{exf@loopdepth}}\exf@tmp{}%
  \addtocounter{exf@loopdepth}{-1}}
```

\exerciseloopstr Loop through list and save result to \exerciseloopret (or optional argument #1):

```
\newcommand{\exerciseloopstr}[3][\exerciseloopret]{%
  \def#1{}\exerciseloop{#2}{\protected@edef#1{#1#3}}}
```

Lists Management.

\exf@notedata@sheet Store a sheet tag:

```
\def\exf@sheetlist{}
\newcommand{\exf@notedata@sheet}[2]{%
  \xdef\exf@sheetlist{\exf@sheetlist{#1}}}
```

exf@notedata@problem Store a problem tag:

```
\def\exf@problemlist{}
\newcommand{\exf@notedata@problem}[2]{%
  \xdef\exf@problemlist{\exf@problemlist{#1}}%
  \if @#2@\else%
   \ifcsname exf@problemlist@#2\endcsname\else%
    \exf@csdo\gdef{exf@problemlist@#2}{}\fi%
   \exf@csdo\xdef{exf@problemlist@#2}%
    {\csname exf@problemlist@#2\endcsname{#1}}%
  \fi}
```

@notedata@subproblem Store a subproblem tag:

```
\newcommand{\exf@notedata@subproblem}[2]{%
  \ifcsname exf@subproblemlist@#2\endcsname\else%
   \exf@csdo\gdef{exf@subproblemlist@#2}{}\fi%
  \exf@csdo\xdef{exf@subproblemlist@#2}%
   {\csname exf@subproblemlist@#2\endcsname{#1}}}
```

\getsheetlist Get list of sheet tags, problem tags (all, within current or particular sheet), subproblem tags
\getproblemlist (within current or particular problem):
\getsubproblemlist

```
\newcommand{\getsheetlist}[1]{\exf@sheetlist}
\newcommand{\getproblemlist}[1]{\if @#1@%
  \ifdefined\sheettag\exf@csor{exf@problemlist@\sheettag}{}%
  \else\exf@problemlist\fi%
 \else%
  \if *#1\exf@problemlist\else\exf@csor{exf@problemlist@#1}{}\fi%
 \fi}
\newcommand{\getsubproblemlist}[1]{\if @#1@%
  \exf@csor{exf@subproblemlist@\problemtag}{}\else%
  \exf@csor{exf@subproblemlist@#1}{}\fi}
```

C.10 Sheet Environment

exf@sheet Define options for `sheet` environment:

```
\define@key{exf@sheet}{points}{\exf@scanpoints\exf@sheet@points#1++@}
\define@key{exf@sheet}{number}{\setcounter{\exf@sheetcounter}{#1}%
  \addtocounter{\exf@sheetcounter}{-1}\refstepcounter{\exf@sheetcounter}}
\define@key{exf@sheet}{label}{\def\exf@label{#1}}
\define@key{exf@sheet}{tag}{\def\sheettag{#1}}
```

sheet Define `sheet` environment (potentially using custom name):

```
\newenvironment{\exf@sheetname}[1][]{%
```

Insert hook code to clear page, step counter:

```
  \exf@config@insertsheetclearpage%
  \refstepcounter{\exf@sheetcounter}%
```

Use equation counter for sheets:

```
  \ifexf@style@sheetequation%
   \exf@eqsav\value{equation}\relax%
   \setcounter{equation}{\value{exf@sheetequation}}%
   \let\theequation\theexf@sheetequation%
   \let\theHequation\theHexf@sheetequation%
  \fi%
```

Reset optional arguments, process arguments:

```
\let\exf@sheet@points\@undefined%
\def\sheettag{\getexerciseconfig{tagsheet}}%
\let\exf@sheet@points@total\@undefined%
\let\exf@label\@undefined%
\setkeys{exf@sheet}{#1}%
```

Process automatic and manual labels:

```
\ifexf@autolabelsheet\label{\exf@config@labelsheet{\sheettag}}\fi%
\ifdefined\exf@label\label{\exf@label}\fi%
\exf@writedata{sheet}{\sheettag}{}%
```

Set points from explicit input or from `.aux` storage:

```
\ifdefined\exf@sheet@points%
 \let\exf@sheet@points@given\exf@empty%
\else%
 \let\exf@sheet@points@given\@undefined%
 \ifcsname exf@sheetpoints@\sheettag\endcsname%
  \exf@csdotwo\let\exf@tmp{exf@sheetpoints@\sheettag}%
  \exf@exptwo\exf@scanpoints\exf@sheet@points\exf@tmp++@%
\fi\fi%
```

Process metadata:

```
\exf@ifis\exf@metadata{sheet}{{%
 \ifx\exf@data@sheet@author\exf@empty\else%
  \let\exf@data@author\exf@data@sheet@author\fi%
 \def\exf@data@title{\exf@config@composemetasheet%
  {\csname the\exf@sheetcounter\endcsname}{\exf@data@sheet@rawtitle}}%
 \exf@writemetadata}\gdef\exf@metadata{off}}%
```

Insert hook code:

```
\exf@config@insertsheetbefore%
```

Add table of contents line:

```
\ifx\exf@config@toclevelsheet\exf@empty\else%
 \ifdefined\phantomsection\phantomsection\fi\fi%
\exf@addcontentsline{\exf@config@toclevelsheet}%
 {\exf@config@composetocsheet{\csname the\exf@sheetcounter\endcsname}%
  {\exf@data@sheet@rawtitle}}%
```

Write sheet title:

```
\exf@config@insertsheettitle}%
```

End of environment; perform sanity check on total points if given explicitly:

```
{\ifdefined\exf@sheet@points@given%
  \exf@warnmismatch{\exf@sheetname}{\exf@sheetcounter}%
   {\exf@sheet@points@total}{\exf@sheet@points}%
```

Test whether points have changed since last compile:

```
\else%
 \exf@warnrerun{\exf@sheetname}{\exf@sheetcounter}%
  {\exf@sheet@points@total}{\exf@sheet@points}%
```

Store points:

```
1508    \let\exf@sheet@points\exf@sheet@points@total%
1509   \fi%
```

Write sheet points to `.aux` file:

```
1510   \ifdefined\exf@sheet@points%
1511    \expandafter\exf@writesheetpoints\exf@sheet@points%
1512   \fi%
```

Insert solutions:

```
1513   \exf@ifis\exf@solutionbelow{sheet}{\insertsolutions}%
```

Insert hook code:

```
1514   \exf@config@insertsheetafter%
1515   \exf@config@insertsheetclearpage%
```

Restore original equation counter:

```
1516   \ifexf@style@sheetequation%
1517    \setcounter{exf@sheetequation}{\value{equation}}%
1518    \setcounter{equation}{\exf@eqsav}%
1519   \fi%
```

Done:

```
1520   \ignorespacesafterend}
```

`rciseclear doublepage` Clear the current page, clear even page with a totally empty page:

```
1521 \newcommand{\exerciseclear doublepage}{%
1522   \clearpage\ifexf@twoside\ifodd\value{page}\else%
1523   \thispagestyle{empty}\hbox{}\newpage\fi\fi}
```

C.11 Problem Environment

Print Problems.

`exf@problem` Define options for `problem` environment:

```
1524 \define@key{exf@problem}{points}{\exf@scanpoints\exf@problem@points#1++@}
1525 \define@key{exf@problem}{label}{\def\exf@label{#1}}
1526 \define@key{exf@problem}{tag}{\def\problemtag{#1}}
1527 \define@key{exf@problem}{sollabel}{\xdef\exf@sollabel{#1}}
```

`printproblem` Define `printproblem` environment:

```
1528 \newenvironment{printproblem}[1]{%
```

Start with new paragraph, set text style, add vspace:

```
1529   \par\exf@config@styletext\addvspace{\exf@config@skipproblemabove}%
```

Step problem counter:

```
1530   \refstepcounter{\exf@problemcounter}%
```

Insert hook code:

```
1531   \exf@config@insertproblembefore%
```

Begin inner group, mark in problem:

```
\begingroup%
\def\exf@in@problem{}%
```

Use equation counter for problems:

```
\ifexf@style@problemequation%
 \exf@eqsav\value{equation}\relax%
 \setcounter{equation}{\value{exf@problemequation}}%
 \let\theequation\theexf@problemequation%
 \let\theHequation\theHexf@problemequation%
\fi%
```

Initialise variables, process arguments:

```
\exf@init@block{\exf@config@skipprobleminfo}%
\def\problemtag{\getexerciseconfig{tagproblem}}%
\let\exf@problem@points\@undefined%
\let\exf@label\@undefined%
\global\let\exf@sollabel\@undefined%
\let\exf@problem@points@total\@undefined%
\setkeys{exf@problem,exf@probleminfo,exf@scanproblem}{#1}%
```

Process automatic and manual labels:

```
\ifexf@autolabelproblem\label{\exf@config@labelproblem{\problemtag}}\fi%
\ifdefined\exf@label\label{\exf@label}\fi%
\exf@writedata{problem}{\problemtag}{\ifdefined\sheettag\sheettag\fi}%
```

Mark for new solution section; remember problem counter, title:

```
\global\let\exf@problem@solfirst\exf@empty%
\xdef\exf@prevprob{\csname the\exf@problemcounter\endcsname}%
\ifcsname theH\exf@problemcounter\endcsname%
 \xdef\exf@prevprobhref{\exf@problemcounter.%
  \csname theH\exf@problemcounter\endcsname}%
\fi%
\ifx\exf@data@problem@rawtitle\exf@empty%
 \global\let\exf@prevprobtitle\@undefined%
\else%
 \protected@xdef\exf@prevprobtitle{\exf@data@problem@rawtitle}%
\fi%
\exf@clear@prevsubprob%
```

Set points from explicit input or from `.aux` storage:

```
\ifdefined\exf@problem@points%
 \let\exf@problem@points@given\exf@empty%
\else%
 \let\exf@problem@points@given\@undefined%
 \ifcsname exf@problempoints@\problemtag\endcsname%
  \exf@csdotwo\let\exf@tmp{exf@problempoints@\problemtag}%
  \exf@exptwo\exf@scanpoints\exf@problem@points\exf@tmp++@%
\fi\fi%
\global\let\exf@prevpoints\exf@problem@points%
\let\exf@problem@points@show\@undefined%
\ifdefined\exf@problem@points%
 \let\exf@problem@points@show\exf@problem@points%
\fi%
```

Disable points display if desired:

```
\exf@ifis\exf@pointsat{off}{\let\exf@problem@points@show\@undefined}%
```

Display points in opening line if desired; expand points into argument and remove points:

```
\exf@ifis\exf@pointsat{start}{\exf@outpoints{\exf@append@intro}%
 {\exf@config@composepointspairstartproblem}{\exf@problem@points@show}}%
\exf@ifis\exf@pointsat{start*}{\exf@outpoints{\exf@prepend@intro}%
 {\exf@config@composepointspairstartproblem}{\exf@problem@points@show}}%
```

Insert hook code, set problem body style:

```
\exf@config@insertprobleminfo%
\exf@config@styletextproblem%
```

Write title without item:

```
\ifdim\exf@config@skipproblemitem=0pt%
 \exf@prepend@intro{{%
  \exf@config@styletitle\exf@config@styletitleproblem%
  \exf@config@composetitleproblem{\csname the\exf@problemcounter\endcsname}%
   {\exf@data@problem@rawtitle}}}%
```

Write item with fixed total width or item width plus space:

```
\else%
 \ifdim\exf@config@skipproblemitem>0pt%
  \setlength\exf@addmargin{\exf@config@skipproblemitem}%
 \else%
  \settowidth\exf@addmargin{%
   \exf@config@styletitle\exf@config@styletitleproblem%
   \exf@config@composeitemproblem{\exf@config@counterproblemmax}%
   \exf@config@composeitemproblemsep}%
 \fi%
```

Define item label:

```
 \def\exf@introitem{\makebox[0cm][r]{%
  \exf@config@styletitle\exf@config@styletitleproblem%
  \exf@config@composeitemproblem{\csname the\exf@problemcounter\endcsname}%
   \exf@config@composeitemproblemsep}}%
```

Compose title:

```
 \ifx\exf@data@problem@rawtitle\exf@empty\else%
  \exf@prepend@intro{{%
   \exf@config@styletitle\exf@config@styletitleproblem%
   \exf@config@composetitleproblem{\exf@empty}{\exf@data@problem@rawtitle}}}%
 \fi%
\fi%
```

Write points into margin if desired; expand points into argument and remove points:

```
\exf@ifis\exf@pointsat{margin}{%
 \exf@outpoints{\exf@prepend@def\exf@introitem}%
 {\exf@makepointsmargin}{\exf@problem@points@show}}%
```

Write out opening line:

```
\exf@open@block{\exf@config@skipproblemtitle}%
```

Add table of contents line:

```
\exf@addcontentsline{\exf@config@toclevelproblem}%
 {\exf@config@composetocproblem{\csname the\exf@problemcounter\endcsname}%
  {\exf@data@problem@rawtitle}}%
```

Done:

```
\@afterindentfalse}%
```

End environment, show points if desired:

```
{\exf@ifis\exf@pointsat{end}{\showpoints}%
```

Perform sanity checks on total points if given explicitly:

```
\ifdefined\exf@problem@points@given%
 \exf@warnmismatch{\exf@problemname}{\exf@problemcounter}%
  {\exf@problem@points@total}{\exf@problem@points}%
```

Warn if calculated total points have changed:

```
\else%
 \exf@warnrerun{\exf@problemname}{\exf@problemcounter}%
  {\exf@problem@points@total}{\exf@problem@points}%
```

Read computed total points:

```
 \let\exf@problem@points\exf@problem@points@total%
\fi%
```

Write points to `.aux` file; add to sheet total:

```
\ifdefined\exf@problem@points%
 \expandafter\exf@writeproblempoints\exf@problem@points%
 \exf@exptwo\exf@addtopoints\exf@sheet@points@total\exf@problem@points%
```

Warn if no points given for present problem but previously:

```
\else\ifdefined\exf@sheet@points@total%
 \PackageWarning{exframe}{no points defined for \exf@problemname}%
\fi\fi%
```

Solutions to subproblems must be declared within problem environment:

```
\exf@clear@prevsubprob%
```

End paragraph and environment:

```
\par\exf@close@block%
```

Display solution if desired:

```
\exf@ifis\exf@solutionbelow{problem}{%
 \exf@config@insertproblemsolution%
 \exf@showsolutions{\exf@config@composetitlesolutionmulti}{}}%
```

Restore original equation counter:

```
\ifexf@style@problemequation%
 \setcounter{exf@problemequation}{\value{equation}}%
 \setcounter{equation}{\exf@eqsav}%
\fi%
```

End inner group:

```
\endgroup%
```

Insert hook code, vertical skip:

```
\exf@config@insertproblemafter%
\addvspace{\exf@config@skipproblembelow}%
```

Display solution if desired:

```
  \exf@ifis\exf@solutionbelow{problem*}{%
   \exf@showsolutions{\exf@config@composetitlesolutionmulti}{}}%
```

Done:

```
  \ignorespacesafterend}
```

Read Problem Environment.

`exf@scanproblem` Define options for scanning `problem` environment:

```
\define@boolkey{exf@scanproblem}[exf@scanproblem@]{disable}[true]{}
```

`exf@problem@direct` Define direct output version of `problem` environment; pass on to `printproblem` environment:

```
\newenvironment{exf@problem@direct}[1][]%
 {\printproblem{#1}}{\endprintproblem\ignorespacesafterend}
```

`exf@problem@scan` `exf@scanproblem` Define scan version of `problem` environment; use `\exf@scanblock` to properly parse optional argument and pass on to `exf@scanproblem`:

```
\newenvironment{exf@problem@scan}%
 {\exf@scanblock{\exf@scanproblem}}{\endexf@scanproblem}%
\newenvironment{exf@scanproblem}[2]{%
```

Determine problem display:

```
  \global\exf@scanproblem@disablefalse%
  \setkeys*{exf@scanproblem}{#1}%
  \exf@config@insertproblemselect{#1}%
  \ifexf@scanproblem@disable\global\exf@scanproblem@disabletrue\fi%
```

Circumvent problem display:

```
  \ifexf@scanproblem@disable%
   \def\exf@verbatim@process{\@gobble}%
  \else%
```

Write separator and `printproblem` environment to buffer:

```
   \ifexf@lineno\exf@addline\exf@probbuf{\exf@linesep}%
    \exf@addline\exf@probbuf{\exf@lineno}\fi%
   \exf@addline\exf@probbuf%
    {\@backslashchar begin{printproblem}{#1}}%
```

Start verbatim processing:

```
   \def\exf@verbatim@process{\exf@append@buf\exf@probbuf}%
  \fi%
  \exf@verbatim#2}%
```

End verbatim processing; close `printproblem` environment:

```
 {\exf@endverbatim%
  \ifexf@scanproblem@disable\else%
   \exf@addline\exf@probbuf{\@backslashchar end{printproblem}}%
   \global\exf@probbuf@cleanfalse%
  \fi%
```

Write buffer to file if output file open:

```
\ifexf@probfile@open%
 \exf@write@buf\exf@probfile\exf@probbuf%
 \exf@clear@probbuf%
\fi%
```

Output buffer immediately:

```
\ifexf@problemmanual\else%
 \exf@source@buf\exf@probbuf%
 \exf@clear@probbuf%
\fi%
```

Done:

```
\ignorespacesafterend}
```

`problem` Define `problem` environment (potentially using custom name) to choose between direct and buffered version:

```
\newenvironment{\exf@problemname}%
 {\ifexf@problembuf\let\exf@tmp\exf@problem@scan%
  \else\let\exf@tmp\exf@problem@direct\fi%
  \exf@tmp}%
 {\ifexf@problembuf\let\exf@tmp\endexf@problem@scan%
  \else\let\exf@tmp\endexf@problem@direct\fi%
  \exf@tmp}
```

C.12 Problem Blocks

Problem Block Handling.

`\exf@problemstitle` Compose the title for a problem block section:

```
\newcommand{\exf@problemstitle}{%
```

Check whether title is empty:

```
 \protected@edef\exf@problemstitleexp{\exf@config@composetitleproblems}%
 \ifx\exf@problemstitleexp\exf@empty\else%
```

Output section line:

```
  \exf@section{\exf@config@skipproblemstitle}%
   {\exf@config@styletitle\exf@config@styletitleproblems%
    \exf@problemstitleexp}%
  \exf@addcontentsline{\exf@config@toclevelproblems}%
   {\exf@config@composetocproblems}%
 \fi}
```

`\exf@showproblemsin` Output problem block intro:

```
\newcommand{\exf@showproblemsin}{%
```

Set problem body style; add vertical space; insert hook code:

```
 \par\exf@config@styletext\addvspace{\exf@config@skipproblemsabove}%
 \exf@config@insertproblemsbefore}
```

`\exf@showproblemsout` Output problem block outro:

```
\newcommand{\exf@showproblemsout}{%
```

Insert hook code; close paragraph; add vertical space:

```
\exf@config@insertproblemsafter%
\par\exf@config@styletext\addvspace{\exf@config@skipproblemsbelow}}
```

`\exf@showproblems` Output problem block in buffer:

```
\newcommand{\exf@showproblems}{%
```

Do nothing if buffer is empty (avoid titles):

```
\ifexf@probbuf@clean\else\begingroup%
```

Execute output problem block intro:

```
\exf@showproblemsin%
\exf@problemstitle%
```

Source and clear buffer:

```
\exf@source@buf\exf@probbuf%
\exf@clear@probbuf%
```

Execute output problem block outro:

```
\exf@showproblemsout%
\endgroup\fi}
```

Problems Buffer Interface.

`\writeproblems` Open a file `#1.prb` for writing problems; default is present main file name:

```
\newcommand{\writeproblems}[1][\jobname]{%
\exf@close@probfile\exf@start@probfile{#1}}
```

`\closeproblems` Close problems output file (if open):

```
\newcommand{\closeproblems}{\exf@close@probfile}
```

`\readproblems` Read problems from file `#1.prb`; default is present main file name; switch layout and add heading:

```
\newcommand{\readproblems}[1][\jobname]{\exf@close@probfile%
\begingroup%
 \exf@config@styletext\exf@config@styletextproblem%
 \exf@problemstitle%
 \input{#1\exf@config@extproblems}%
\endgroup}
```

`\insertproblems` Show problems buffer:

```
\newcommand{\insertproblems}{\exf@showproblems}
```

C.13 Subproblem Environment

Print Subproblems.

`exf@subproblem` Define options for `subproblem` environment:

```
\define@key{exf@subproblem}{points}{\exf@scanpoints\exf@subproblem@points#1++@}
\define@key{exf@subproblem}{label}{\def\exf@label{#1}}
\define@key{exf@subproblem}{tag}{\def\subproblemtag{#1}}
```

printsubproblem Define `printsubproblem` environment:

```
\newenvironment{printsubproblem}[1]{%
```

Start with new paragraph, set text style, add vspace and step counter:

```
  \par{\exf@config@styletext\addvspace{\exf@config@skipsubproblemabove}}%
  \refstepcounter{\exf@subproblemcounter}%
```

Insert hook code:

```
  \exf@config@insertsubproblembefore%
```

Start inner group, mark in subproblem:

```
  \begingroup%
  \def\exf@in@subproblem{}%
```

Initialise variables, process arguments:

```
  \exf@init@block{\exf@config@skipsubprobleminfo}%
  \def\subproblemtag{\getexerciseconfig{tagsubproblem}}%
  \let\exf@subproblem@points\@undefined%
  \let\exf@label\@undefined%
  \setkeys{exf@subproblem,exf@probleminfo,exf@scansubproblem}{#1}%
```

Process automatic and manual labels:

```
  \ifexf@autolabelproblem\label{\exf@config@labelsubproblem%
   {\subproblemtag}}\fi%
  \ifdefined\exf@label\label{\exf@label}\fi%
  \exf@writedata{subproblem}{\subproblemtag}{\problemtag}%
```

Remember subproblem counter for solution:

```
  \xdef\exf@prevsubprob{\csname the\exf@subproblemcounter\endcsname}%
  \ifcsname theH\exf@subproblemcounter\endcsname%
   \xdef\exf@prevsubprobhref{\exf@subproblemcounter.%
    \csname theH\exf@subproblemcounter\endcsname}%
  \fi%
```

Remember points for display; disable points display if desired:

```
  \let\exf@subproblem@points@show\@undefined%
  \ifdefined\exf@subproblem@points%
   \let\exf@subproblem@points@show\exf@subproblem@points\fi%
  \exf@ifis\exf@subpointsat{off}{\let\exf@subproblem@points@show\@undefined}%
```

Write points to `.aux` file; add to problem total:

```
  \ifdefined\exf@subproblem@points%
   \global\let\exf@prevpoints\exf@subproblem@points%
   \expandafter\exf@writesubproblempoints\exf@subproblem@points%
   \exf@exptwo\exf@addtopoints\exf@problem@points@total\exf@subproblem@points%
```

Warn if no points given for present subproblem but previously:

```
  \else\ifdefined\exf@problem@points@total%
   \PackageWarning{exframe}{no points defined for \exf@subproblemname}%
  \fi\fi%
```

Display points in opening line if desired; expand points into argument and remove points:

```
  \exf@ifis\exf@subpointsat{start}{\exf@outpoints{\exf@append@intro}%
   {\exf@config@composepointspairstartsubproblem}{\exf@subproblem@points@show}}%
```

```
\exf@ifis\exf@subpointsat{start*}{\exf@outpoints{\exf@prepend@intro}%
 {\exf@config@composepointspairstartsubproblem}{\exf@subproblem@points@show}}%
```

Insert hook code:

```
\exf@config@insertsubprobleminfo%
```

Write opening line without item:

```
\ifdim\exf@config@skipsubproblemitem=0pt%
 \exf@prepend@intro{{%
  \exf@config@styletitle\exf@config@styletitlesubproblem%
  \exf@config@composetitlesubproblem{%
   \csname the\exf@subproblemcounter\endcsname}}}%
```

Write item with fixed total width or item width plus space:

```
\else%
 \ifdim\exf@config@skipsubproblemitem>0pt%
  \setlength\exf@addmargin{\exf@config@skipsubproblemitem}%
 \else%
  \settowidth\exf@addmargin{%
   \exf@config@styletitle\exf@config@styletitlesubproblem%
   \exf@config@composeitemsubproblem{\exf@config@countersubproblemmax}%
   \exf@config@composeitemsubproblemsep}%
 \fi%
```

Define item label:

```
 \def\exf@introitem{\makebox[0cm][r]{%
  \exf@config@styletitle\exf@config@styletitlesubproblem%
  \exf@config@composeitemsubproblem%
   {\csname the\exf@subproblemcounter\endcsname}%
  \exf@config@composeitemsubproblemsep}}%
\fi%
```

Write points into margin if desired; expand points into argument and remove points:

```
\exf@ifis\exf@subpointsat{margin}{%
 \exf@outpoints{\exf@prepend@def\exf@introitem}%
 {\exf@makepointsmargin}{\exf@subproblem@points@show}}%
```

Write out opening line:

```
\exf@open@block{\exf@config@skipsubproblemtitle}%
```

Done:

```
\@afterindentfalse}%
```

End environment, show points if desired:

```
{\exf@ifis\exf@subpointsat{end}{\showpoints}%
```

End paragraph and environment:

```
\par\exf@close@block%
```

Display solution if desired:

```
\exf@ifis\exf@solutionbelow{subproblem*}{%
 \exf@config@insertsubproblemsolution%
 \exf@showsolutions{\exf@config@composetitlesolutionsingle}{}}%
```

End inner group:

```
\endgroup%
```

Insert hook code, vertical skip:

```
\exf@config@insertsubproblemafter%
{\exf@config@styletext\addvspace{\exf@config@skipsubproblembelow}}%
```

Display solution if desired:

```
\exf@ifis\exf@solutionbelow{subproblem}{%
 \exf@showsolutions{\exf@config@composetitlesolutionsingle}{}}%
```

Done:

```
\ignorespacesafterend}
```

Read Subproblem Environment.

`exf@scansubproblem` Define options for scanning `subproblem` environment:

```
\define@boolkey{exf@scansubproblem}[exf@scansubproblem@]{disable}[true]{}
```

`xf@subproblem@direct` Define direct output version of `subproblem` environment; pass on to `printsubproblem` environment:

```
\newenvironment{exf@subproblem@direct}[1][]%
 {\printsubproblem{#1}}{\endprintsubproblem\ignorespacesafterend}
```

`exf@subproblem@scan` `exf@scansubproblem` Define scan version of `subproblem` environment; use `\exf@scanblock` to properly parse optional argument and pass on to `exf@scansubproblem`:

```
\newenvironment{exf@subproblem@scan}%
 {\exf@scanblock{\exf@scansubproblem}}{\endexf@scansubproblem}%
\newenvironment{exf@scansubproblem}[2]{%
```

Determine subproblem display:

```
\global\exf@scansubproblem@disablefalse%
\setkeys*{exf@scansubproblem}{#1}%
\exf@config@insertsubproblemselect{#1}%
\ifexf@scansubproblem@disable\global\exf@scansubproblem@disabletrue\fi%
```

Circumvent subproblem display:

```
\exf@clear@subprobbuf%
\ifexf@scansubproblem@disable%
 \global\let\exf@prevsubprob\exf@empty%
 \def\exf@verbatim@process{\@gobble}%
\else%
```

Write `printsubproblem` environment to buffer:

```
 \exf@addline\exf@subprobbuf%
  {\@backslashchar begin{printsubproblem}{#1}}%
```

Start verbatim processing:

```
 \def\exf@verbatim@process{\exf@append@buf\exf@subprobbuf}%
\fi%
\exf@verbatim#2}%
```

End verbatim processing; close `printsubproblem` environment:

```
{\exf@endverbatim%
 \ifexf@scansubproblem@disable\else%
  \exf@addline\exf@subprobbuf{\@backslashchar end{printsubproblem}}%
 \fi%
```

Output buffer immediately:

```
 \exf@source@buf\exf@subprobbuf%
```

Done:

```
 \ignorespacesafterend}
```

`subproblem` Define `subproblem` environment (potentially using custom name) to choose between direct and buffered version:

```
\newenvironment{\exf@subproblemname}%
 {\ifexf@subproblembuf\let\exf@tmp\exf@subproblem@scan%
  \else\let\exf@tmp\exf@subproblem@direct\fi%
  \exf@tmp}%
 {\ifexf@subproblembuf\let\exf@tmp\endexf@subproblem@scan%
  \else\let\exf@tmp\endexf@subproblem@direct\fi%
  \exf@tmp}
```

C.14 Solution Environment

Print Solutions.

`exf@solution` Define options for `solution` environment:

```
\define@key{exf@solution}{prob}{\def\exf@solprob{#1}}
\define@key{exf@solution}{subprob}{\def\exf@solsubprob{#1}}
\define@key{exf@solution}{problemtag}{\def\problemtag{#1}}
\define@key{exf@solution}{sheettag}{\def\sheettag{#1}}
\define@key{exf@solution}{href}{\def\exf@solhref{#1}}
\define@key{exf@solution}{label}{\def\exf@label{#1}}
\define@key{exf@solution}{points}{\exf@scanpoints\exf@solution@points#1++@}
\define@key{exf@solution}{probtitle}{\def\exf@solprobtitle{#1}}
\define@key{exf@solution}{firstsol}[]{\let\exf@solfirst\exf@empty}
```

`printsolution` Define `printsolution` environment to display a previously read `solution` environment; this works analogously to `problem` and `subproblem`:

```
\newenvironment{printsolution}[1]{%
```

Start new paragraph, add vertical space:

```
 \par{\exf@config@styletext\addvspace{\exf@config@skipsolutionabove}}%
```

Insert hook code:

```
 \exf@config@insertsolutionbefore%
```

Use equation counter for solutions:

```
 \ifexf@style@solutionequation%
  \exf@eqsav\value{equation}\relax%
  \setcounter{equation}{\value{exf@solutionequation}}%
  \let\theequation\theexf@solutionequation%
```

```
  \let\theHequation\theHexf@solutionequation%
 \fi%
```

Start a group, initialise variables, process arguments:

```
 \begingroup%
 \def\exf@in@solution{}%
 \def\exf@solprob{}%
 \def\exf@solsubprob{}%
 \let\exf@solfirst\@undefined%
 \let\exf@label\@undefined%
 \let\exf@solution@points\@undefined%
 \let\exf@solution@points@total\@undefined%
 \def\exf@solhref{}%
 \exf@init@block{\exf@config@skipsolutioninfo}%
 \setkeys{exf@solution,exf@probleminfo,exf@scansolution}{#1}%
```

Set solution counter to reflect associated problem:

```
 \exf@csdo\def{the\exf@solutioncounter}%
  {\exf@config@composeitemsolutionlabel{\exf@solprob}{\exf@solsubprob}}%
 \refstepcounter{\exf@solutioncounter}%
```

Set label:

```
 \ifdefined\exf@label\label{\exf@label}\fi%
```

Remember points for display; disable points display if desired:

```
 \let\exf@solution@points@show\@undefined%
 \ifdefined\exf@solution@points%
  \let\exf@solution@points@show\exf@solution@points\fi%
 \exf@ifis\exf@solpointsat{off}{\let\exf@solution@points@show\@undefined}%
```

Display points in opening line if desired; expand points into argument and remove points:

```
 \exf@ifis\exf@solpointsat{start}{\exf@outpoints{\exf@append@intro}%
  {\exf@config@composepointspairstartsolution}{\exf@solution@points@show}}%
 \exf@ifis\exf@solpointsat{start*}{\exf@outpoints{\exf@prepend@intro}%
  {\exf@config@composepointspairstartsolution}{\exf@solution@points@show}}%
```

Insert hook code, set solution body style:

```
 \exf@config@insertsolutioninfo%
 \exf@config@styletext\exf@config@styletextsolution%
```

Determine solution for problem or subproblem:

```
 \ifx\exf@solsubprob\exf@empty%
  \let\exf@tmp\exf@config@skipsolutionitem%
 \else%
  \let\exf@tmp\exf@config@skipsolutionitemsub%
 \fi%
```

Write title without item:

```
 \ifdim\exf@tmp=0pt%
  \protected@edef\exf@solution@title{%
   \exf@composetitle{\exf@solprob}{\exf@solsubprob}}%
  \ifx\exf@solsubprob\exf@empty\ifdefined\exf@solfirst\else%
   \let\exf@solution@title\exf@empty\fi\fi%
  \ifx\exf@solution@title\exf@empty\else%
   \exf@prepend@intro{{%
```

```
    \exf@config@styletitle\exf@config@styletitlesolution%
    \ifexf@solutionhref\exf@href{\exf@solhref}%
     {\exf@solution@title}\else\exf@solution@title\fi}}%
  \fi%
```

Write item with fixed total width or item width plus space:

```
  \else%
   \ifdim\exf@tmp>0pt%
    \setlength\exf@addmargin{\exf@tmp}%
   \else%
    \settowidth\exf@addmargin{%
     \exf@config@styletitle\exf@config@styletitlesolution%
     \ifx\exf@solsubprob\exf@empty%
      \let\exf@tmp\exf@config@composeitemsolutionfirst\else%
      \let\exf@tmp\exf@config@composeitemsolutionfirstsub\fi%
     \protected@edef\exf@solution@item{\exf@tmp%
      {\exf@config@counterproblemmax}{\exf@config@countersubproblemmax}}%
     \ifx\exf@solution@item\exf@empty\else%
      \exf@solution@item\exf@config@composeitemsolutionsep\fi}%
   \fi%
```

Set item label depending on problem or subproblem:

```
   \ifx\exf@solsubprob\exf@empty\ifdefined\exf@solfirst%
     \let\exf@tmp\exf@config@composeitemsolutionfirst\else%
     \let\exf@tmp\exf@config@composeitemsolution\fi%
   \else\ifdefined\exf@solfirst%
     \let\exf@tmp\exf@config@composeitemsolutionfirstsub\else%
     \let\exf@tmp\exf@config@composeitemsolutionsub\fi\fi%
   \protected@edef\exf@solution@item{\exf@tmp{\exf@solprob}{\exf@solsubprob}}%
```

Define item label:

```
   \def\exf@introitem{\ifx\exf@solution@item\exf@empty\else\makebox[0cm][r]{%
    \exf@config@styletitle\exf@config@styletitlesubproblem%
    \ifexf@solutionhref\exf@href{\exf@solhref}{\exf@solution@item}%
    \else\exf@solution@item\fi%
    \exf@config@composeitemproblemsep}\fi}%
  \fi%
```

Write points into margin if desired; expand points into argument and remove points:

```
   \exf@ifis\exf@solpointsat{margin}{%
    \exf@outpoints{\exf@prepend@def\exf@introitem}%
    {\exf@makepointsmargin}{\exf@solution@points@show}}%
```

Write out opening line:

```
   \exf@open@block{\exf@config@skipsolutiontitle}%
```

Done:

```
   \@afterindentfalse}%
```

End environment, show points if desired, perform sanity check:

```
  {\exf@ifis\exf@solpointsat{end}{\showpoints}%
   \exf@warnmismatch{\exf@solutionname}{\exf@solutioncounter}%
    {\exf@solution@points@total}{\exf@solution@points}%
```

End paragraph and envionment:

```
   \par\exf@close@block%
```

Restore original equation counter:

```
\ifexf@style@solutionequation%
 \setcounter{exf@solutionequation}{\value{equation}}%
 \setcounter{equation}{\exf@eqsav}%
\fi%
```

End inner group:

```
\endgroup%
```

Vertical skip, insert hook code:

```
{\exf@config@styletext\addvspace{\exf@config@skipsolutionbelow}}%
\exf@config@insertsolutionafter%
```

Done:

```
\ignorespacesafterend}
```

`\solutionssection` Define a section for a problem within a solution block:

```
\newcommand{\solutionssection}[1]{\begingroup%
```

Initialise variables, process arguments:

```
\def\exf@solprob{}%
\def\exf@solsubprob{}%
\def\exf@solprobtitle{}%
\let\exf@label\@undefined%
\let\exf@solhref\@undefined%
\setkeys{exf@solution,exf@scansolution}{#1}%
```

Select title (and table of contents entry) corresponding to multiple problems vs. single problem:

```
\let\exf@composesectitle\exf@config@composetitlesolutionsproblemmulti%
\exf@ifis\exf@solutionbelow{problem}{\let\exf@composesectitle%
  \exf@config@composetitlesolutionsproblemsingle}%
\exf@ifis\exf@solutionbelow{problem*}{\let\exf@composesectitle%
  \exf@config@composetitlesolutionsproblemsingle}%
\def\exf@solutionstoc{\exf@addcontentsline{\exf@config@toclevelsolution}%
  {\exf@config@composetocsolution{\exf@solprob}{\exf@solprobtitle}}}%
```

Write section line:

```
\addvspace{\exf@config@skipsolutionsproblemabove}%
\exf@solutionssection{\exf@config@styletitlesolutionsproblem}%
 {\exf@composesectitle{\exf@solprob}{\exf@solprobtitle}}%
 {\exf@config@skipsolutionsproblemtitle}%
 {\exf@solutionstoc}{\exf@label}{\exf@solhref}%
\endgroup}
```

Read Solution Environment.

`exf@scansolution` Define options for scanning `solution` environment:

```
\define@key{exf@scansolution}{forproblem}[]{\exf@clear@prevsubprob}
\define@boolkey{exf@scansolution}[exf@scansolution@]{disable}[true]{}
```

`xf@clear@prevsubprob` Clear previous subproblem register:

```
\newcommand{\exf@clear@prevsubprob}{%
  \global\let\exf@prevsubprob\@undefined%
  \global\let\exf@prevsubprobhref\@undefined%
  \global\let\exf@prevpoints\@undefined%
  \global\exf@scansubproblem@disablefalse}
```

`\exf@process@solsec` If this is the first solution within a problem section, write section heading to buffer:

```
\newcommand{\exf@process@solsec}{%
  \def\exf@probarg{\ifdefined\exf@prevprob prob={\exf@prevprob}\fi%
   \ifdefined\exf@prevprobtitle,probtitle={\exf@prevprobtitle}\fi%
   \ifdefined\exf@prevprobhref,href={\exf@prevprobhref}\fi%
   \ifdefined\exf@sollabel,label={\exf@sollabel}\fi}%
  \exf@ifis\exf@solutionbelow{here}{\let\exf@probarg\@undefined}%
  \exf@ifis\exf@solutionbelow{subproblem}{\let\exf@probarg\@undefined}%
  \exf@ifis\exf@solutionbelow{subproblem*}{\let\exf@probarg\@undefined}%
  \ifdefined\exf@probarg%
   \ifexf@lineno\exf@addline\exf@solbuf{\exf@linesep}%
    \exf@addline\exf@solbuf{\exf@lineno}\fi%
   \exf@addline\exf@solbuf{\@backslashchar solutionssection{\exf@probarg}}%
   \exf@addline\exf@solbuf{}%
  \fi}
```

`@generate@solprobarg` Declare additional arguments to `printsolution` to describe corresponding problem and tags:

```
\newcommand{\exf@generate@solprobarg}{%
  \edef\exf@solprobarg{%
    \ifdefined\exf@problem@solfirst firstsol,\fi%
    \ifdefined\exf@prevprob prob={\exf@prevprob},\fi%
    \ifdefined\exf@prevsubprob subprob={\exf@prevsubprob},%
     \ifdefined\exf@prevsubprobhref href={\exf@prevsubprobhref},\fi%
    \else%
     \ifdefined\exf@prevprobhref href={\exf@prevprobhref},\fi%
    \fi%
    \ifdefined\exf@prevpoints points=%
     {\expandafter\exf@formatpoints\exf@prevpoints},\fi%
    \ifdefined\sheettag sheettag={\sheettag},\fi%
    \ifdefined\problemtag problemtag={\problemtag},\fi}}
```

`exf@solution@direct` Define direct output version of `solution` environment; pass on to `printsolution` environment:

```
\newenvironment{exf@solution@direct}[1][]%
 {\showpoints%
  \exf@generate@solprobarg%
  \global\let\exf@problem@solfirst\@undefined%
  \exf@clear@prevsubprob%
  \exf@showsolutionsin%
  \let\exf@composetitle\exf@config@composetitlesolutionsingle%
  \exf@exptwo\printsolution{\exf@solprobarg#1}}%
 {\endprintsolution%
  \exf@showsolutionsout%
  \ignorespacesafterend}
```

`exf@solution@scan` `exf@scansolution` Define scan version of `solution` environment; use `\exf@scanblock` to properly parse optional argument and pass on to `exf@scansolution`:

```
\newenvironment{exf@solution@scan}%
```

```
 {\exf@scanblock{\exf@scansolution}}{\endexf@scansolution}%
\newenvironment{exf@scansolution}[2]{%
```

Determine solution association and display:

```
  \exf@scansolution@disablefalse%
  \setkeys*{exf@scansolution}{#1}%
  \ifdefined\exf@prevsubprob%
   \ifexf@scansubproblem@disable\exf@scansolution@disabletrue\fi%
  \else%
   \ifexf@scanproblem@disable\exf@scansolution@disabletrue\fi%
  \fi%
```

Circumvent solution display:

```
  \ifexf@scansolution@disable%
   \exf@clear@prevsubprob%
   \def\exf@verbatim@process{\@gobble}%
  \else%
```

If solution is to be displayed immediately, make sure to display points first; insert solution section heading in buffer; compose additional arguments to `printsolution`:

```
   \exf@ifis\exf@solutionbelow{here}{\showpoints}%
   \exf@generate@solprobarg%
   \ifdefined\exf@problem@solfirst\exf@process@solsec\fi%
   \global\let\exf@problem@solfirst\@undefined%
   \exf@clear@prevsubprob%
```

Write separator and `printsolution` environment to buffer:

```
   \ifexf@lineno\exf@addline\exf@solbuf{\exf@linesep}%
    \exf@addline\exf@solbuf{\exf@lineno}\fi%
   \exf@addline\exf@solbuf%
    {\@backslashchar begin{printsolution}{\exf@solprobarg#1}}%
```

Start verbatim processing:

```
   \def\exf@verbatim@process{\exf@append@buf\exf@solbuf}%
  \fi%
  \exf@verbatim#2}%
```

End verbatim processing; close `printsolution` environment:

```
 {\exf@endverbatim%
  \ifexf@scansolution@disable\else%
   \exf@addline\exf@solbuf{\@backslashchar end{printsolution}}%
   \global\exf@solbuf@cleanfalse%
  \fi%
```

Write buffer to file if output file open:

```
  \ifexf@solfile@open%
   \exf@write@buf\exf@solfile\exf@solbuf%
   \exf@clear@solbuf%
  \fi%
```

Drop buffer if solutions not to be displayed:

```
  \ifsolutions\else\exf@clear@solbuf\fi%
```

Display solution immediately in various cases:

```
  \exf@ifis\exf@solutionbelow{here}{%
```

```
    \exf@showsolutions{\exf@config@composetitlesolutionsingle}{}}%
   \ifdefined\exf@in@subproblem\else%
    \exf@ifis\exf@solutionbelow{subproblem}{%
     \exf@showsolutions{\exf@config@composetitlesolutionsingle}{}}%
    \exf@ifis\exf@solutionbelow{subproblem*}{%
     \exf@showsolutions{\exf@config@composetitlesolutionsingle}{}}\fi%
   \ifdefined\exf@in@problem\else%
    \exf@ifis\exf@solutionbelow{problem}{%
     \exf@showsolutions{\exf@config@composetitlesolutionmulti}{}}%
    \exf@ifis\exf@solutionbelow{problem*}{%
     \exf@showsolutions{\exf@config@composetitlesolutionmulti}{}}\fi%
```

Done:

```
   \ignorespacesafterend}
```

`solution` Define `solution` environment (potentially using custom name) to choose between direct and buffered version:

```
\newenvironment{\exf@solutionname}%
 {\ifexf@solutionbuf\let\exf@tmp\exf@solution@scan%
  \else\let\exf@tmp\exf@solution@direct\fi%
  \exf@tmp}%
 {\ifexf@solutionbuf\let\exf@tmp\endexf@solution@scan%
  \else\let\exf@tmp\endexf@solution@direct\fi%
  \exf@tmp}
```

C.15 Solution Blocks

Solution Block Handling.

`exf@solutionssection` Output solution block section:

```
\newcommand{\exf@solutionssection}[6]{%
```

Check whether title is empty:

```
   \protected@edef\exf@solutionstitleexp{#2}%
   \ifx\exf@solutionstitleexp\exf@empty\else%
```

Define a label:

```
    \ifdefined#5%
     \exf@csdo\def{the\exf@solutioncounter}%
      {\exf@config@composeitemsolutionlabel{\exf@solprob}{\exf@solsubprob}}%
     \refstepcounter{\exf@solutioncounter}\label{#5}%
    \fi%
```

Output section line:

```
    \exf@section{#3}{\exf@config@styletitle\exf@config@styletitlesolution#1%
     \ifexf@solutionhref\exf@href{#6}{\exf@solutionstitleexp}%
     \else\exf@solutionstitleexp\fi}#4%
   \fi}
```

`\exf@solutionstitle` Compose the title for a solution block:

```
\newcommand{\exf@solutionstitle}{\exf@solutionssection%
  {\exf@config@styletitlesolutions}{%
   \exf@config@composetitlesolutions}{\exf@config@skipsolutionstitle}%
```

```
    {\exf@addcontentsline{\exf@config@toclevelsolutions}%
     {\exf@config@composetocsolutions}}{\@undefined}{\@undefined}}
```

`\exf@showsolutionsin` Output solution block intro:

```
\newcommand{\exf@showsolutionsin}{%
```

Set solution body style; add vertical space; insert hook code:

```
  \par\exf@config@styletext\addvspace{\exf@config@skipsolutionsabove}%
  \exf@config@styletextsolution%
  \exf@config@insertsolutionsbefore}
```

`exf@showsolutionsout` Output solution block outro:

```
\newcommand{\exf@showsolutionsout}{%
```

Insert hook code; close paragraph; add vertical space:

```
  \exf@config@insertsolutionsafter%
  \par\exf@config@styletext\addvspace{\exf@config@skipsolutionsbelow}}
```

`\exf@showsolutions` Output solution block in buffer:

```
\newcommand{\exf@showsolutions}[2]{%
```

Do nothing if buffer is empty (avoid titles):

```
  \ifexf@solbuf@clean\else\begingroup%
```

Execute output solution block intro:

```
    \exf@showsolutionsin%
    \let\exf@composetitle#1%
    #2%
```

Source and clear buffer:

```
    \exf@source@buf\exf@solbuf%
    \exf@clear@solbuf%
```

Execute output solution block outro:

```
    \exf@showsolutionsout%
  \endgroup\fi}
```

Solutions Buffer Interface.

`\writesolutions` Open a file `#1.sol` for writing solutions; default is present main file name:

```
\newcommand{\writesolutions}[1][\jobname]{%
  \exf@close@solfile\exf@start@solfile{#1}}
```

`\closesolutions` Close solutions output file (if open):

```
\newcommand{\closesolutions}{\exf@close@solfile}
```

`\readsolutions` Read solutions from file `#1.sol`; default is present main file name; switch layout and add heading:

```
\newcommand{\readsolutions}[1][\jobname]{\exf@close@solfile%
  \ifsolutions\begingroup%
```

```
    \exf@config@styletext\exf@config@styletextsolution%
    \let\exf@composetitle\exf@config@composetitlesolutionmulti%
    \exf@solutionstitle%
    \input{#1\exf@config@extsolutions}%
  \endgroup\fi}
```

`\insertsolutions` Show solutions buffer:

```
\newcommand{\insertsolutions}{\exf@showsolutions%
  {\exf@config@composetitlesolutionmulti}{\exf@solutionstitle}}
```

C.16 Interaction with metastr

This package transfers some functionality to the package metastr when loaded previously (preferably with package option `course`) by the package option `metastr`. The following changes apply:

- The terms specified by the configuration options `term...`, see section C.3, are transferred to the corresponding term registers `\metaterm{...}` in metastr. This can facilitate internationalisation, and suitable words for English, German, French and Spanish are provided.
- Metadata as described in section 2.3 should be filled via `\metaset` rather than `\exercisedata`. The present package will read it from the metastr registers.
- Basic pdf metadata is written automatically or manually as described in section 2.3. This is done via `\metawritepdfinfo`, consequently, automatic writing of these basic metadata is disabled in metastr.
- Sheet-specific metadata (package option `pdfdata=sheet`) is handled via the metastr registers `sheettitle` and `sheetdata` rather than `composemetasheet`.

Apply modifications only when package metastr is loaded:

```
\ifdefined\metaset
```

Transfer Metadata. Import basic data from metastr:

```
\exercisedata{author={\metapick[]{author}}}
\exercisedata{title={\metapick[]{title}}}
\exercisedata{subject={\metapick[]{subject}}}
\exercisedata{keyword={\metapick[]{keyword}}}
\exercisedata{date={\metapick[]{date}}}
```

Import course data from metastr:

```
\ifdefined\mstr@def@course
\exercisedata{course={\metapick[]{course}}}
\exercisedata{instructor={\metapick[]{instructor}}}
\exercisedata{institution={\metapick[]{institution}}}
\exercisedata{period={\metapick[]{period}}}
\exercisedata{material={\metapick[]{material}}}
\fi
```

Sheet Data.

`sheettitle` `sheetauthor` Registers for sheet title and author:

```
\metadef{sheettitle}
\metadef{sheetauthor}
```

`\exf@writemetadata` Write pdf metadata via metastr package, let exframe initiate process (especially for sheet):

```
\metaunset[info]{writepdf}
\def\exf@writemetadata{%
  \exf@ifis\exf@metadata{sheet}{\metaset[use]{sheettitle}{}}%
  \metawritepdfinfo%
  \exf@ifis\exf@metadata{sheet}{\metaunset[use]{sheettitle}}}
```

Fill registers:

```
\metaset{sheetauthor}{\exerciseifempty{\getsheetdata{author}}%
  {\metapick[#1]{instructor}}{\getsheetdata{author}}}
\metaset{sheettitle}{\exerciseifempty{\getsheetdata{rawtitle}}%
  {\metatranslate[#1]{sheet} \thesheet}%
  {\getsheetdata{rawtitle}}}
\metaset{author}{\exerciseifempty{\getsheetdata{author}}%
  {\metapick[#1]{instructor}}{\metapick[#1]{sheetauthor}}}
\metaset{subtitle}{%
  \metaif[use]{sheettitle}%
   {\metapick[#1]{sheettitle}}%
   {\metapick[#1]{material}}}
```

Terms.

`term...` Transfer term definitions to metastr:

```
\exerciseconfig{termsheet}{\metaterm{sheet}}
\exerciseconfig{termsheets}{\metaterm{sheets}}
\exerciseconfig{termproblem}{\metaterm{problem}}
\exerciseconfig{termproblems}{\metaterm{problems}}
\exerciseconfig{termsolution}{\metaterm{solution}}
\exerciseconfig{termsolutions}{\metaterm{solutions}}
\exerciseconfig{termpoint}{\metaterm{point}}
\exerciseconfig{termpoints}{\metaterm{points}}
```

Translations. English:

```
\ifdefined\mstr@lang@en
\metasetterm[en]{sheet}{Sheet}
\metasetterm[en]{sheets}{Sheets}
\metasetterm[en]{problem}{Problem}
\metasetterm[en]{problems}{Problems}
\metasetterm[en]{solution}{Solution}
\metasetterm[en]{solutions}{Solutions}
\metasetterm[en]{point}{point}
\metasetterm[en]{points}{points}
\fi
```

German:

```
\ifdefined\mstr@lang@de
\metasetterm[de]{sheet}{Blatt}
\metasetterm[de]{sheets}{Blätter}
\metasetterm[de]{problem}{Aufgabe}
\metasetterm[de]{problems}{Aufgaben}
\metasetterm[de]{solution}{L\"osung}
```

```
\metasetterm[de]{solutions}{L\"osungen}
\metasetterm[de]{point}{Punkt}
\metasetterm[de]{points}{Punkte}
\fi
```

French:

```
\ifdefined\mstr@lang@fr
\metasetterm[fr]{sheet}{Feuille}
\metasetterm[fr]{sheets}{Feuilles}
\metasetterm[fr]{problem}{Probl\‘eme}
\metasetterm[fr]{problems}{Probl\‘emes}
\metasetterm[fr]{solution}{Solution}
\metasetterm[fr]{solutions}{Solutions}
\metasetterm[fr]{point}{point}
\metasetterm[fr]{points}{points}
\fi
```

Spanish:

```
\ifdefined\mstr@lang@es
\metasetterm[es]{sheet}{Hoja}
\metasetterm[es]{sheets}{Hojas}
\metasetterm[es]{problem}{Problema}
\metasetterm[es]{problems}{Problemas}
\metasetterm[es]{solution}{Solucion}
\metasetterm[es]{solutions}{Soluciones}
\metasetterm[es]{point}{punto}
\metasetterm[es]{points}{puntos}
\fi
```

Close metastr conditional:

```
\fi
```