The exesheet class and package

Antoine Missier
antoine.missier@ac-toulouse.fr

2024/12/22, v2.8

Contents

1 Introduction

The exesheet *package* is designed for typesetting exercise or exam sheets. Additionally, the exesheet *class* loads the schooldocs package [1]. The latter makes adjustments to margins and titles, and defines various layout styles with specific headers and footers suitable for exercise sheets, among other uses. Refer to the documentation of the schooldocs package for more details. The exesheet *class* is build upon the article class and forwards any unknown options to it.

There are many other packages dedicated to exercise sheets. In section 6.3 we provide an overview of some of their functionalities. Most of them suggest encapsulating each exercise within an environment. In contrast, exesheet starts each exercise with `\exercise`, which functions similarly to a subsection (with the same features) and is suitable for documents that primarily consist of exercises. The package also offers alternative ways to introduce exercises, which are more appropriate for shorter exercises.

Another distinctive feature of the exesheet package is its specific settings for enumeration lists, which are suitable for numbering questions or answers within an exercise.

For all exercises within the sheet, you can display only the questions, only the answers, or both, all while preserving their placement as they appear in the source file. This choice allows for great flexibility: you can create a correct version for all exercises collectively, or individual corrections per exercise, per part (subpart of exercise), per question, per sub-question.

The ability to hide questions or answers is found in many packages, but the main interest of exesheet is to be able to display or not a detailed scoring guide, along with correction instructions. This is very useful for grading papers with multiple graders. Furthermore exesheet can check the consistency of the scale.

Many settings can be customized, and various options are available to manage the output document. These options rely on the key-val mechanism: `key=value`. These options can be applied when calling the class or the package, e.g.

```
\documentclass[a4paper,11pt,output=answers,display=pts]{exesheet}
```

or later using the command `\exesheetset{`⟨*options*⟩`}`. In the example above, `a4paper,11pt` are options that are passed to the underlying article class.

In the current document, a frame is utilized to emphasize output examples.

2 Titles

2.1 The `\exercise` command

`\exercise` The `\exercise[`⟨*opt*⟩`]` command initiates an exercise with the title **Exercise**, typeset as a document subsection, followed by automatic numbering, unique to the entire document. The optional parameter ⟨*opt*⟩ is utilized to include additional text on the same title line, such as specifying a subject or a marking scheme. Thus, using `\exercise[(to begin)]` results in:

Exercise 1 (to begin)

Try this first command; easy!

To bring optional text closer to the exercise number, you can employ `\unskip` which removes any preceding space. Take a look at the following example, achieved with `\exercise[\unskip*** (difficult)]`:

> **Exercise 2*** (difficult)**
>
> Calculate $1 + 1$.

`\exercisename` The word "Exercise" is automatically translated into various languages[1] depending on the language that is loaded (via babel or polyglossia). You can alter it by modifying `\exercisename`. A better approach is to use macros from the translations package by Clemens Niederberger [7] (which allows language switching), e.g. `\DeclareTranslation{swedish}{exesheet-exercise}{\"Ovning}`.

`\labelexercise` This command combines `\exercisename` with the exercise number and can be redefined. For instance, if you want to include a period after the exercise number, you can redefine it as follows:
`\renewcommand{\labelexercise}{\exercisename~\theexercise.}`

`\theexercise` If you wish to alter only the numbering style, you can redefine `\theexercise` which is based on the `exercise` counter.

`\labelexercisestyle` This macro, which is initially empty, enables the definition of a specific style for exercise titles. In this document, we have set the following in the preamble: `\renewcommand{\labelexercisestyle}{\rmfamily\color{black}}`[2].

`\exercise*` The starred version `\exercise*[`⟨*opt*⟩`]{`⟨*label*⟩`}` permits the selection of an alternative ⟨*label*⟩ for a specific exercise while omitting the numbering. For instance: `\exercise*[(Fermat's theorem)]{Problem}` results in:

> **Problem (Fermat's theorem)**
>
> Prove that there are no positive integers x, y, z such that $x^n + y^n = z^n$ for any integer n greater than 2.

2.2 The `\subpart` command

`\subpart` An exercise may consist of multiple parts, which can be created using the `\subpart[`⟨*opt*⟩`]` command. The part title is typeset similar to a sub-subsection.

> **Exercise 3**
>
> **Part A (preliminary)**
>
> To begin, prepare your cup of tea.
>
> **Part B**
>
> Now you are ready to proceed with the current exercise.

[1]Currently, translation is integrated into the package for the following languages: French, German, Spanish, Italian, and Portuguese.

[2]In this document, real section and subsection titles have been highlighted by modifying their color and font (sans serif) using the `\allsectionsfont` macro from the sectsty package [10].

The following macros allow customization in the same manner as for `\exercise`.

`\thesubpart` By default, subpart numbering employs letters : A, B, C, and so on. This numbering style can be modified using the `\thesubpart` command, which relies on the `subpart` counter. For example, you can redefine it as follows: `\renewcommand\thesubpart{\arabic{subpart}}`.

`\subpartname` `\labelsubpart` `\labelsubpartstyle` The `\subpart` command utilizes `\subpartname` (with automatic translation in several languages according to the chosen language), as well as `\labelsubpart` and `\labelsubpartstyle`, all of which can be modified.

`\subpart*` Similar to `\exercise*`, the starred version `\subpart*[⟨opt⟩]{⟨label⟩}` permits an alternative ⟨*label*⟩ and omits the numbering. For instance, you can use `\subpart*{First part}`.

2.3 The `\annex` command

`\annex` The `\annex[⟨opt⟩]` command composes the title **ANNEX** in uppercase letters, centered, using the subsection style, with an optional parameter that will be added on the same line.

ANNEX (to be returned)

`\annexname` The term “Annex” is automatically translated into several languages (depending on the chosen language). It can be extended to additional languages or altered by redefining `\annexname` or by utilizing macros from the translations package [7].

`\annexstyle` The style of the annex title is determined by the `\annexstyle` macro, which is defined as follows: `\newcommand\annexstyle{\MakeUppercase}`. This command may be redefined according to your preferences.

2.4 Titles in the table of contents

`[exetoc=⟨bool⟩]` By default, the titles **Exercise**, **Part** and **Annex** are included in the table of contents, if there is any, or in the PDF file's summary when the hyperref package is utilized. To prevent this, you can set the package option `exetoc=false` (with the default being `true`). However, note that optional title arguments will always be ignored in the table of contents.

2.5 Short exercises: the `\exe` command

`\exe` The `\exe` command initiates an exercise with the abbreviation **Ex.** followed by the exercise number. This is achieved without utilizing sectioning commands, and the exercise content begins on the same line. An exercise begins a new paragraph without any indentation.

Ex. 4 — This is a brief exercise that can encompass several paragraphs or questions.

Here for example a new paragraph begins.

Ex. 5 — This is another concise exercise.

`\exname` `\exlabel` `\exsepmark` The abbreviation **Ex** can be modified by redefining `\exname` or with macros from the translations package [7]. The `\exlabel` macro combines `\exname` with a period then the exercise number (given by the same `exercise` counter), while `\exsepmark` typesets a long dash. These characteristics can be altered by redefining these commands.

`\exe*` The starred version doesn't display a separator, as demonstrated below:

Ex. 6 Another short exercise without a separator.

3 Enumerations and lists

3.1 List settings

`enumerate` (*env.*) `\item` Enumeration lists are used to represent questions and sub-questions within exercises. To provide clear emphasis, labels are typeset in bold. Additionally, these labels are aligned to the left, positioned at the start of the line without indentation, and the vertical spacing between items is increased compared to standard LaTeX lists. These formatting adjustments are achieved using the `\setlist` command, a feature from the enumitem package by Javier Bezos [3].

Exercise 7

1. First question

  (a) First sub-question

  (b) Second sub-question

2. Second question

The `enumerate` environment takes an optional parameter, that allows, among others things, the typesetting of alternative list labels. For instance, typing `\begin{enumerate}[label=\alph*),font=\itshape\normalfont]` will produce the labels "*a*), *b*), *c*)...". There are many other options available (see the enumitem [3] package documentation)[3]. Label font formatting can be changed globally using `\setlist[enumerate]{font=...}` (called *after* `\begin{document}`).

Lists created with the `itemize` environment retain their default configuration[4].

`[setlist=⟨bool⟩]` The package option `setlist=false` prevents changes to enumeration lists and reverts to the default LaTeX settings (the default value is `true`).

3.2 List of exercises : the `exenumerate` environment

`exenumerate` (*env.*) When an exercise sheet consists of short, independent questions, it might be unreasonable to display the full title **Exercise** for each one. In addition to the previously

[3]Labels can also be modified using a "shortlabel" argument, e.g. `\begin{enumerate}[A.]`, or globally through the redefinition of `\labelenumi` or `\labelenumii` commands.

[4]However, the `french` option of the babel package changes the appearance of `itemize` lists and employs long dashes as labels for each list level. This can cause issues when mathematical content follows the dash symbol, as it might be mistaken for the minus sign. Thus, with the option `setlist=true`, the default LaTeX `itemize` list style is reinstated with `\frenchsetup{StandardLists=true}`.

mentioned `\exe` command, we offer an even more streamlined solution using the `exenumerate` environment. This environment is essentially an enumeration list with increased spacing between items, compared to the `enumerate` environment. Here is an example (the main list uses the `exenumerate` environment, while the sub-list is created using the standard `enumerate` environment):

1. Translate the following sentences in English:

  (a) Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

  (b) Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus.

2. Translate the following sentence in German:

  Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi.

3. Translate the following sentence in French: Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

The `exenumerate` environment (also based on the `enumitem` [3] package) accepts an optional parameter, similar to the `enumerate` environment.

3.3 Items aligned by row: `tablenum1`, `tablenuma`, `tablitem`

`tablenum1` (*env.*)
`tablenuma` (*env.*)
`tablitem` (*env.*)

These three environments are employed to typeset brief questions (`tablenum1`), sub-questions (`tablenuma`) or `itemize` lists (`tablitem`) on the same line. They share the same syntax: `\begin{tablenum1}[`⟨*opt*⟩`](`⟨*cols*⟩`)`. The ⟨*cols*⟩ parameter denotes the number of columns utilized by the environment. It must be enclosed *in parentheses*. This parameter can be omitted, in which case its default value is 2. Similar to conventional lists, each item is initiated with the `\item` command.

Internally we have utilized the `\NewTasksEnvironment` macro from the `tasks` package by Clemens Niederberger [4]. The usage of the optional argument ⟨*opt*⟩ is explained in the documentation of this package. For example, similar to the `enumitem` package [3], `label=\arabic*)` produces an Arabic numbering followed by a closing parenthesis. Additionally there are numerous possibilities for arranging items in original ways. For instance, the `\item*` command allows you to specify the number of columns the item is supposed to span. In the subsequent example, the five `\item` commands are sequentially positioned between `\begin{tablenum1}(3)` and `\end{tablenum1}`. Notice that numbering occurs line by line in this context.

Exercise 8

Calculate the derivative of the following functions:

1. $f(x) = \dfrac{1-x^2}{\mathrm{e}^x + \mathrm{e}^{-x}}$,	**2.** $g(x) = \ln\left(\dfrac{1-x}{1+x^2}\right)$,	**3.** $h(x) = \displaystyle\int_0^1 \mathrm{e}^{xy}\,\mathrm{d}y$,
4. $k(x) = \displaystyle\sum_{i=1}^{\infty} \frac{1}{x^i}$,	**5.** $l(x) = \displaystyle\int_{\frac{1}{x}}^{x} \frac{1}{\ln t}\,\mathrm{d}t$.	

For `tablenuma`, labels are letters, a, b, c, ..., enclosed in parentheses.

`\labelenumone` `\labelenuma` You can change the labels by redefining the macros `\labelenumone` (for `tablenum1`) and `\labelenuma` (for `tablenuma`), using the `task` counter: e.g. `\renewcommand\labelenuma{\Alph{task}.}` yields the labels **A.**, **B.**, etc.

`\enumfont` With the default option `setlist=true`, the font of all enumeration labels may be changed by redefining `\enumfont` (`\bfseries` by default). If the `exesheet` package is invoked with the option `setlist=false`, labels within `tablenum1` and `tablenuma` environments will be presented with indentation, and in normal font rather than bold. You can change the label formatting globally with the command `\settask`, e.g. `\settask{label-format=\itshape}`. You can also completely redefine the environments using `\RenewTasksEnvironment`. When `setlist=true`, place these commands *after* `\begin{document}`.

When you intend to utilize `tablenuma` (or `tablitem`) immediately after inserting the `\item` command within an `enumerate` environment, a vertical misplacement may occur as shown below:

1.
(a) $f(x) = \dfrac{1-x^2}{e^x + e^{-x}}$ **(b)** $g(x) = \ln\left(\dfrac{1-x}{1+x^2}\right)$, **(c)** $h(x) = \displaystyle\int_0^1 e^{xy}\,dy$.

To adjust the vertical alignment, include `\mbox{}\vspace{`⟨*height*⟩`}` just after `\item` and before invoking `\begin{tablenuma}` (or `\begin{tablitem}`), where ⟨*height*⟩ can be a positive or negative length. Here we used `\vspace{-5.4ex}`.

1. **(a)** $f(x) = \dfrac{1-x^2}{e^x + e^{-x}}$ **(b)** $g(x) = \ln\left(\dfrac{1-x}{1+x^2}\right)$, **(c)** $h(x) = \displaystyle\int_0^1 e^{xy}\,dy$.

3.4 Items aligned by column: `colsenum`, `colsitem`

`colsenum` (*env.*) To achieve numbering of items by column, we provide the `colsenum` environment: `\begin{colsenum}[`⟨*opt*⟩`]{`⟨*cols*⟩`}`. The mandatory parameter is the number of columns, and the optional parameter will be passed to the underlying `enumerate` environment, allowing you to change the numbering type (e.g. a, A, etc.), among other possibilities. *To use this environment, you need to load the* multicol *package in the preamble.* Here's an example with `\begin{colsenum}{3}`:

Exercise 9

Calculate the derivative of the following functions:

1. $f(x) = \dfrac{1-x^2}{e^x + e^{-x}}$, **3.** $h(x) = \displaystyle\int_0^1 e^{xy}\,dy$, **5.** $l(x) = \displaystyle\int_{\frac{1}{x}}^{x} \frac{1}{\ln t}\,dt$.

2. $g(x) = \ln\left(\dfrac{1-x}{1+x^2}\right)$, **4.** $k(x) = \displaystyle\sum_{i=1}^{\infty} \frac{1}{x^i}$,

`colsenum*` (*env.*) It may be observed that, on each line, items are not necessarily properly aligned, which can result in ungraceful effects. On the other hand, the `colsenum` environment doesn't attempt to align columns from the bottom by adjusting the vertical spacing between items. If you desire this alignment (which is the default behavior in `multicol`), you can use the `colsenum*` environment (with the same syntax as `colsenum`). Here's what we obtain with `colsenum*`:

Exercise 10

Calculate the derivative of the following functions:

1. $f(x) = \dfrac{1-x^2}{\mathrm{e}^x + \mathrm{e}^{-x}}$, **3.** $h(x) = \displaystyle\int_0^1 \mathrm{e}^{xy}\,\mathrm{d}y$, **5.** $l(x) = \displaystyle\int_{\frac{1}{x}}^{x} \frac{1}{\ln t}\,\mathrm{d}t$.

2. $g(x) = \ln\left(\dfrac{1-x}{1+x^2}\right)$, **4.** $k(x) = \displaystyle\sum_{i=1}^{\infty} \frac{1}{x^i}$,

We can observe that these alignments are not as elegant as those achieved through row numbering. However, column numbering might still be more suitable when dealing with numerous items of varying heights, and especially when the number of items can differ from column to column. Additionally, a benefit of `colsenum` is that the label selection is automatic, based on the list level (and the language), unlike `tablenum1` or `tablenuma`.

`colsitem` (*env.*)
`colsitem*` (*env.*)

For `itemize` lists, the `colsitem` environment generates items aligned by column, unlike the line-by-line alignment of `tablitem`. It follows the same syntax as `colsenum`: `\begin{colsitem}[`⟨*opt*⟩`]{`⟨*cols*⟩`}`. The optional parameter, passed to the underlying `itemize` environment, allow to change the item label (bullet by default). Furthermore, just like `colsenum*`, the `colsitem*` environment produces column alignment from the bottom. *The multicol package is also required and must be loaded in the preamble.*

4 Questions and solutions

4.1 Environments `questions` and `answers`

`questions` (*env.*)
`answers` (*env.*)

The exesheet package offers two environments, `questions` and `answers`, which allow you to optionally show or hide questions and answers within exercises.

`[output=`⟨*opt*⟩`]`

The output is governed by the `output` key option which recognizes three values: `questions`, `answers`, and `both`. The `questions` value shows only questions without answers, `answers` displays answers without questions, and `both` (the default option) displays both questions and answers.

`\correctionstyle`
`correctioncolor`

In the default case where both questions and answers are displayed, the answers are typeset using the `\correctionstyle` style, which utilizes the color `correctioncolor`. You can modify this color using the `\definecolor` macro[5]. By default, `\definecolor{correctioncolor}{rgb}{0,0.2,0.6}` is used, resulting in a kind of dark blue.

`\correctionname`

Furthermore, when using `output=both` the title **Correction** is displayed at the beginning of `answers` environments. This title is defined by the `\correctionname` macro, with translation available in several languages, and it can also be modified. For instance you might prefer “Solution” over “Correction”. The style defined by `\correctionstyle` will be applied to the title as well as the entire environment. Here’s an example to illustrate this:

[5]The `\definecolor` command is provided by the xcolor package developed by Uwe Kern, which is automatically loaded by exesheet.

Exercise 11

1. Is the exesheet package useful ?
2. Aren't there any other packages that deal with exercises ?

Correction

1. The exesheet package is useful for teachers.
2. There are numerous other packages that handle exercises and provide the capability to create questions and solutions separately. For instance the exercise package by Paul Pichaureau, exercises by Roger Jud, exsheets (now superseded by xsim) by Clemens Niederberger, exframe by Niklas Beisert, exam by Philip Hirschhorn, answers by Mike Piff and Joseph Wright, probsoln by Nicola Talbot, eqexam by D. P. Story... They are briefly presented in section 6.3.

When only answers are displayed, the text color remains black and the word "Correction" is not displayed.

4.2 More about `answers` environments

Internally, we have utilized the `\comment` and `\endcomment` macros from the versions package by Uwe Lück [5]. Several other packages also enable selective management of code portions. Notably, the verbatim package by Rainer Schöpf, comment by Victor Eijkhout, version by Donald Arseneau and Stephen Bellantoni, optional by Donald Arseneau and codesection by Matthias Pospiech. Moreover, the versions package [5] offers the `\excludeversion{`⟨*env*⟩`}` and `\includeversion{`⟨*env*⟩`}` macros which allow for the exclusion or inclusion of any environment ⟨*env*⟩. These "optional" environments can be nested[6].

However the `questions` and `answers` environments serve a broader purpose beyond merely displaying or hiding text. You can choose to have a single answers environment for the entire sheet, or alternatively, have separate answers environments for each exercise, exercise part, question, or sub-question. The format in which the title **Correction** should appear in the output, and its placement in the table of contents or PDF file summary, depends on the nesting level of the environment. In fact, the rendering of the **Correction** title and its corresponding table of contents level will be automatically calculated by the environment.

`answers[`⟨*level*⟩`]` (*env.*)

However, users might wish to adjust the title's level themselves. To achieve this, you can manually set the level of the title "Correction" using an optional ⟨*level*⟩ argument which is defined as follows: 1 for section-level titles, 2 for subsections (akin to **Exercise**), 3 for sub-subsections (similar to **Part**), other numbers for lower levels (which won't appear in the table of contents or in the PDF file's summary).

`answers*`

The starred version `answers*` doesn't display the **Correction** title.

[6]The codesection package also supports such nesting, including within the preamble, as well as the optional package, but the latter manages only short sections of optional code.

4.3 Commands `\question`, `\answer` and `\answerspace`

`\question` `\answer` Instead of using `questions` and `answers` environments, we can also employ the simpler `\question{`⟨*ques*⟩`}` and `\answer{`⟨*ans*⟩`}` macros. The visibility of ⟨*ques*⟩ and ⟨*ans*⟩ content is regulated by the same previous `output=`⟨*opt*⟩ key option. This approach might be more fitting when you wish to display answers immediately after each question item. The title "Correction" won't appear at the start of each answer with the `\answer` macro. The answers are also formatted using `\correctionstyle` if `output=both`. However these commands do not support `verbatim` text within them, unlike the `questions` and `answers` environments.

`\question*` `\answer*` When a code must be executed only when questions are displayed but not answers, or the contrary, you have the starred versions e.g. `\question*{\pagebreak}`.

`\answerspace` Some teachers are accustomed to providing their students with documents where questions are typeset, leaving blank spaces instead of answers. This layout allows students to fill in their responses on the paper. Thanks to a suggestion from Maxime Chupin, we achieve this with the `\answerspace{`⟨*height*⟩`}` macro, in which the parameter ⟨*height*⟩ is a valid length, e.g. `\answerspace{3cm}`.

`[answerspace=`⟨*bool*⟩`]` The blank spaces introduced by `\answerspace` can be displayed or hidden, controlled by the `answerspace` option key, which can be set to either `true` or `false` (the default). The `answerspace` key option has no effect (equivalent to `false`) when the answers are displayed (`output=answers` or `both`). Of course the `\answerspace` macro is not meant to be used within `answers` environments.

5 Marking scheme commands

The exesheet package provides several commands to display a marking scheme, with optional comments and explanations about answers in the margins.

5.1 The `\points` command

`\points` The `\points{`⟨*pts*⟩`}` command displays the number of points awarded for an exercise. It is intended to be included in the optional argument of the `\exercise` command[7]. In the following example, we used `\exercise[\points{5}]`:

Exercise 12 5 points

Try to read this document to the end without drinking tea and you get five points.

When only the answers are displayed in an exercise, the `\points` macro doesn't show the points. Further, we provide another macro, which displays points in `questions` like here, and differently in `answers` environments (see section 5.5).

`\pointsname` `\pointname` The term "points" (or "point" in the singular if ⟨*pts*⟩ is less than 2) is appended and is automatically translated into several languages (and can also be modified).

`\pointsstyle` `pointscolor` You can adjust the `\points` command's style through `\pointsstyle`. The color setting (red by default) is managed by `pointscolor` using `\definecolor`, for example you can declare: `\definecolor{pointscolor}{named}{blue}`.

[7] However using `\points` in the optional argument of `\exercise` is not compatible with the memoir class, as the memoir class redefines section commands.

5.2 The `\pts` command

`\pts` When exercises are typeset using the `\exe` macro or as a list with the `exenumerate` environment, the marking scheme can be shown in the margin, aligned with the line where the `\pts{`⟨*num*⟩`}` command is placed (typically the first line of the exercise). The ⟨*num*⟩ parameter represents the number of points assigned to the exercise. Here's an example with `\exe\pts{3}... \exe\pts{1.5}...`

(3 pts) **Ex. 13** — The first short exercise with a marking scheme.

(1.5 pt) **Ex. 14** — The second one.

`\ptsname` `\ptname` `ptscolor` `\ptsstyle` The abbreviation "pts" (or "pt" when the number of points is less than 2) is added automatically using `\ptsname` or `\ptname` macros (translated in several languages if babel or polyglossia is loaded). The point's display color is defined by `ptscolor`, changeable via `\definecolor` (red by default). The display style is determined by `\ptsstyle`, which among other things, adds parenthesis around.

`[display=`⟨*opt*⟩`]` The marking scheme visibility is controlled by the `display` option key. The default option is `display=none`, keeping the marking scheme hidden. To reveal the marking scheme, use `display=pts`. More details are available in section 5.4.

`[marginpos=`⟨*opt*⟩`]` The positioning of the scale is determined by the `marginpos` option key, typically `left` or `right`. The default value is `left` even though LaTeX positions marginal notes on the right side by default. This option has no impact when `display=none`.

For a two-sided document, the default behavior is to place text in the outer margin, which is wider than the inner margin (that contains the binding). The outer margin is positioned on the right side on odd pages and on the left side on even pages. Therefore, the `marginpos` option can also take the values `inner` or `outer`. If you specify `left` or `right` when the `twoside` mode is activated, this value will be converted to `outer`, accompanied by a warning message.

With the `twoside` mode, marginal notes might occasionally appear on the wrong side of a page. This is a known LaTeX bug, and the solution involves using the mparhack package by Tom Sgouros and Stefan Ulrich [9] (which exesheet automatically includes for documents in two-side mode) and *running LaTeX twice*. If necessary, a warning message will prompt you to perform the re-run.

5.3 Commands `\totalexe`, `\note*` and `\note`

For a more comprehensive marking scheme, the following commands are available.

`\totalexe` The `\totalexe{`⟨*num*⟩`}` macro displays the total number of points of an exercise. By default, it appears inside an oval box, with the addition of the word "pts" (or "pt") in bold red. In the following example, the exercise title has been generated using `\exercise[\totalexe{4}]`.

`\note*` For each answer or solution in the correct version, the `\note*{`⟨*num*⟩`}` command indicates the number of points allocated to that question. The appearance slightly varies compared to `\pts`: by default the number is displayed in bold without the "pts" or "pt" suffix, and without parenthesis. In the following example, for answer 3, we employed `\note*{1.5}`, placed right after `\item`.

`\note` The `\note{`⟨*comment*⟩`}` macro is utilized to provide additional information regarding the marking scheme and to explain how points are assigned. In the ⟨*comment*⟩ argument you can use `\\` to create a line break or even `\\[`⟨*height*⟩`]` to adjust the line spacing by ⟨*height*⟩.

`\note[`⟨*num*⟩`]` Placing `\note*{`⟨*num*⟩`}\note{`⟨*comment*⟩`}` at the beginning of an answer is often practical. In such cases LaTeX will align the margin notes vertically, which leads to a warning like: `LaTeX Warning: Marginpar on page ... moved.` However, this warning is not an issue, as LaTeX can usually handle the arrangement of these marginal notes, stacking them one below the other. Nonetheless, to prevent unnecessary warnings, you can combine both commands into a single one by specifying the number of points as an optional argument of the `\note` command: `\note[`⟨*num*⟩`]{`⟨*comment*⟩`}`.

The first comment in the following example is obtained using (immediately after `\item`) `\note[1]{0.5 for the anti-derivative\\0.5 for simplifying}`.

4 pts

Exercise 15

For each subsequent question, determine whether the statement is true or false. Provide a thorough justification for your answer.

1. $\displaystyle\int_0^{\sqrt{3}} \frac{1}{x+\sqrt{3}}\,\mathrm{d}x = \ln 2,$ **2.** $\displaystyle\int_2^{\mathrm{e}} \frac{1}{x\ln x}\,\mathrm{d}x = -\ln 2,$

3. The function F, defined on $\mathbf{R}$ by $F(x) = \displaystyle\int_0^x \frac{1}{t^2+t+1}\,\mathrm{d}t$, is increasing on $\mathbf{R}$.

Correction

1
0.5 for the anti-derivative
0.5 for simplifying

1. We calculate:

$$\int_0^{\sqrt{3}} \frac{1}{x+\sqrt{3}}\,\mathrm{d}x = \Big[\ln\left(x+\sqrt{3}\right)\Big]_0^{\sqrt{3}} = \ln\left(2\sqrt{3}\right) - \ln\sqrt{3} = \ln\left(\frac{2\sqrt{3}}{\sqrt{3}}\right) = \ln 2.$$

TRUE.

1.5
1 for the anti-derivative
0.5 for the final value

Other method: $\frac{1}{x\ln x} > 0$ on $[2,\mathrm{e}]$ whereas $-\ln 2 < 0$

2. We have $\dfrac{1}{x\ln x} = \dfrac{\frac{1}{x}}{\ln x} = \dfrac{u'(x)}{u(x)}$ with $u(x) = \ln x$, which is positive on $[2,\mathrm{e}]$. Hence

$$\int_2^{\mathrm{e}} \frac{1}{x\ln x}\,\mathrm{d}x = \Big[\ln(\ln x)\Big]_2^{\mathrm{e}} = \ln(\ln\mathrm{e}) - \ln(\ln 2) = \ln 1 - \ln(\ln 2) = -\ln(\ln 2).$$

FALSE.

1.5

3. The function F, defined on $\mathbf{R}$ by

$$F(x) = \int_0^x \frac{1}{t^2+t+1}\,\mathrm{d}t,$$

0.5 for F'
1 for the sign of F' and conclusion

is derivable on $\mathbf{R}$ and its derivative is such that $F'(x) = \frac{1}{x^2+x+1}$. The denominator is a quadratic polynomial, always positive because its discriminant is $\Delta = -3 < 0$. Thus F is increasing on $\mathbf{R}$.
TRUE.

In the comment for answer 2, a larger vertical space is created with the optional argument `\\[2ex]` for line break. The last comment, which isn't positioned next to the points number, was produced by placing the following on the first line after the formula: `\note{0.5 for $F'$\\1 for the sign of $F'$ and conclusion}`.

`markingcolor` `\markingstyle` `\ptsboxlength` The color and style for displaying points in `\totalexe` and `\note*` can be customized using `markingcolor` and `\markingstyle`, respectively. The oval box produced by `\totalexe` is created using the `\ovalbox` command from the fancybox package by Timothy Van Zandt [6], with corner arcs set by `\cornersize{1}`. The box's length is determined by `\ptsboxlength`, and not by the box's content, to ensure uniformity across exercises.

`notecolor` `\notestyle` By default, comment notes are typeset in a dark green color defined by `\definecolor{notecolor}{rgb}{0.0,0.4,0.0}`. The style of comments is determined by the `\notestyle` macro.

5.4 Margin notes options

`[display=`⟨*opt*⟩`]` The `display` key option governs the presentation of the marking scheme: as discussed previously (subsection 5.2), `display=none` shows nothing. When using `display=pts` the numbers provided as arguments to `\pts`, `\totalexe`, `\note*` or as optional arguments of `\note[`⟨*num*⟩`]{...}` will be exhibited. The final option is `display=notes` which reveals the complete marginal notes, containing points and comments (the mandatory argument of `\note`), as illustrated in the previous example.

`[marginpos=`⟨*opt*⟩`]` As previously mentioned in subsection 5.2, the side on which to position the scale is determined by the `marginpos` key option, with possible values of `left` and `right` (or `inner` and `outer` if the document is in `twoside` mode).

`[marginwidth=`⟨*opt*⟩`]` The margin layout is governed by the `marginwidth` key option, which can take one of the following values: `standard`, `expand`, or `unset`.

This option has no effect when `display=none`. In this case, both the left and right margins have the same width, except in a two-sided document where the ratio between the left and right margins is 2:3. Otherwise the `marginwidth` key option behaves as follows:

`standard` The left margin is widened, and the right margin is reduced, with a ratio of 3:2 (or 2:3 if `marginpos=right`). The text body is shifted without changing its width. The margin paragraph width remains relatively short (depends on page geometry). This option is not ideal for lengthy comments.

`expand` (default value) The behavior is the same as with the `standard` value when `display=pts`. However, when `display=notes`, the margin expands with a ratio of 3:1 (or 1:3) and the width of margin paragraphs increases.

`unset` This option is provided for cases where the previous settings are not suitable. In this case, no adjustments are made to the margin width. Instead, you can define your own settings using the convenient `\geometry` macro from the geometry package by Hideo Umeki [2]. For instance, you can place the following in the preamble:

```
\geometry{hmarginratio=2:1,marginparwidth=2.5cm}.
```

If `marginpos=right`, you need to invert the ratio, e.g. 1:2 instead of 2:1. If `marginwidth` is not set to `unset`, such a command will have no effect.

Margin settings are applicable to the entire document and need to be configured in the preamble.

`[noteragged=⟨opt⟩]` The package option `noteragged` controls the text alignment within the margins for the mandatory argument of `\note`. It offers the following values: `left`, `right`, `center`, `justify` or `twoside`. The default value is `noteragged=left`, resulting in right-aligned text, which is common for text in the left margin. When `noteragged=right`, the text is left-aligned. Using `justify` makes the text justified, aligning with LaTeX's default behavior for marginal notes. Finally `noteragged=twoside` aligns text to the left on odd pages and to the right on even pages in a two-sided document. It has no effect otherwise (the default `noteragged=left` is used and a warning message appears in the terminal).

When `display` is not set to `notes`, the `noteragged` option has no impact, as it specifically applies to text within the mandatory argument of `\note`.

5.5 The `\totalpoints` command

`\totalpoints` The `\totalpoints{⟨num⟩}` macro serves as a replacement for `\points` when using a comprehensive marking scheme. When the scale is not displayed, it functions similarly to `\points` (visible in questions but not in answers), and when the scale is shown, it's akin to `\totalexe`. For instance, in exercise 15, we could have used `\totalpoints` instead of `\totalexe`. Thus, if the detailed marking scheme is not displayed, the total points would be presented similarly to exercise 5.1.

5.6 Marking scheme consistency checking

`[checkpts=⟨bool⟩]` The marking scheme can be checked out[8] using the key-val option `checkpts=true` (or just `checkpts`); the default value is `false`.

For each exercise, the cumulative points allocated to each question (via `\pts`, `\note*` or `\note[ ]` are compared to the exercise's total specified in `\points`, `\totalexe` or `\totalpoints`. A warning message will be displayed in the shell to indicate whether the scale is valid for the exercise or not. For example:

```
Package exesheet warning: Exercise 3: Sum of points is 4.5pt
                         instead of 5pt.
```

Both comma notation (e.g. 4,5) and decimal point format (e.g. 4.5) may be accepted, depending on your chosen language. The control is made at the beginning of the subsequent exercise, inside the `\points`, `\totalexe` or `\totalpoints` macros. No deep checking will be processed at this level if no points are displayed for the questions inside the exercise (with `display=none` option).

`\totalsheet` At the end of the document, the last exercise is checked, followed by a global examination of the entire sheet. This last task requires knowledge of the total points for the sheet, which must be given by the `\totalsheet{⟨points⟩}` macro in the preamble; otherwise, a warning message will be displayed. If subtotals have been assigned to exercises and *displayed*, the overall comparison is made between the sum of these subtotals and the total points recorded using `\totalsheet`. If not, the evaluation encompasses the sum of points for each individual question. A subsequent warning message indicates the outcome of this last verification. Finally, a message indicates whether all scale controls have been successfully passed or not.

[8]Thanks to Denis Bitouzé for his suggestion about this feature.

6 Options and comparison with other packages

6.1 Summary of available options

Here we provide a summary table of the available options. Details on their usage can be found in the respective sections. The default value is displayed in bold.

Key	Possible values	See section
`exetoc`	**`true`**, `false`	2.4
`setlist`	**`true`**, `false`	3.1
`output`	`questions`, `answers`, **`both`**	4.1
`answerspace`	`true`, **`false`**	4.3
`display`	**`none`**, `pts`, `notes`	5.2, 5.4
`marginpos`	**`left`** `(inner)`, `right` `(outer)`	5.2, 5.4
`marginwidth`	`standard`, **`expand`**, `unset`	5.4
`noteragged`	**`left`**, `right`, `center`, `justify`, `twoside`	5.4
`checkpts`	`true`, **`false`**	5.6
`correct`	`true`, **`false`**, `conditional`	see below

When an invalid key is provided, an error is generated. However, an unrecognized value only triggers a warning message:

```
Value ... is not supported by ... option on input line ...
```

For each option, you can set them through the class or package invocation, e.g. `\usepackage[output=answers,display=notes,noteragged=right]{exesheet}`

`\exesheetset` You can also use the `\exesheetset`{list of ⟨*key*⟩=⟨*value*⟩} command. Note that some options, `output`, `answerspace`, `display`, and `noteragged`, can be changed dynamically, even within the document, while the others are applicable in the preamble exclusively. Dynamic options are processed with each call, whereas the others are processed once, at the beginning of the document.

`[correct=`⟨*opt*⟩`]` A special option, `correct`, can be employed when using the exesheet *class* or in conjunction with the schooldocs package. This option adds "Correct version" (or its translation) to the document title and headers. Possible values are: `true`, `false` (by default) or `conditional`. Using `correct=conditional`, it behaves as `true` when answers are displayed and `false` when they're not.

6.2 Alternative commands

Prior to version 2.0, we used specialized commands to configure output and display options. We have now implemented *key=value* options. Although the latter are more user-friendly, one may prefer the old commands, so they are still supported, but will trigger a warning message. These commands are presented below.

However, the previous options `nosetlist` and `notoc` are no longer supported.

`\questionsonly` `\answersonly` The command `\questionslonly` is equivalent to setting `output=questions` and `\answersonly` means `output=answers`.

`\displaypts` `\displaypoints` The commands `\displaypts` and `\displaypoints` are equivalent to setting `display=pts`.

`\displaynotes`
`\displaynotesright`

`\displaynotes` means `display=notes`, and `\displaynotesright` corresponds to `display=notes,marginpos=right`. These two commands have an optional argument `\displaynotes{⟨ragged⟩}` where ⟨*ragged*⟩ is an alignment command to work inside margin notes. By default it is `\RaggedLeft` with `\displaynotes` and `RaggedRight`[9] with `\displaynotesright`.

6.3 Comparison with other packages

In this section, we will provide an overview of the functionalities (when version 2.7 of this package was published, at February 13, 2024) of various packages or classes found in the 'Exercise' or 'Exam' sections of the CTAN archives (Comprehensive TEX Archive Network). Considering the substantial number of packages in these sections, some omissions may have been unintentionally made. Those excluded are those with documentation not in English or primarily dedicated to producing multiple-choice questions or random question generation. We have focused here on typesetting functionalities and not on managing exercise databases as there are specialized packages or external softwares for that.

The following table is not a result of tests but presents a summary of information collected from the documentation of these packages.

A. exercise, Paul Pichaureau [11]
B. exercises, Roger Jud [12]
C. xsim, Clemens Niederberger [13]
D. exframe, Niklas Beisert [14]
E. exam, Philip Hirschhorn [15]
F. answers, Mike Piff and Joseph Wright [16]
G. probsoln, Nicola L.C. Talbot [17]
H. exsol, Walter Daems [18]
I. exercisepoints, Henning Kerstan [19]
J. worksheet, Benjamin Zöllner [20]
K. exam-n, Norman Gray [21]
L. eqexam, D. P. Story [22]
M. cesenaexam, Alex Pacini [23]
N. esami, Grazia Messineo, Salvatore Vassallo [24]
O. randexam, Jianrui Lyu [25]
P. hideanswer, Yukoh Kusakabe [26]
Q. mathexam, Jan Hlavacek [27]
R. exesheet, Antoine Missier

Functionality	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Optional text for exercise titles	✘		✘	✘	✘				✘				✘					✘
Subparts of exercises	✘			✘	✘				✘		✘	✘	✘	✘	✘			✘
Annex title or appendix															✘			✘
Exercise titles in TOC of PDF files	✘																	✘
Short labels for exercises	✘		✘	✘														✘
Hiding questions or answers*	✘	*	✘	✘	*	✘	✘	✘			*	✘		✘	*	*		✘
Different placements for answers	✘										✘	✘						✘
Change answers placement in output	✘			✘		✘		✘				✘						
Blank spacing in place of answers		✘	✘		✘							✘		✘	✘		✘	✘
Marking scheme commands		✘	✘	✘	✘				✘	✘	✘	✘		✘	✘			✘
Various positions of points				✘	✘							✘		✘				✘
Marking scheme calculation/checking		✘	✘	✘	✘				✘		✘	✘			✘			✘
Detailed notes for scoring guide																		✘

[9]These commands come from the ragged2e package by Martin Schröder [8].

7 Implementation

7.1 Options and required packages

The `exesheet` class is build upon the `article` class and transfers all its unknown options to it. The use of `\ProcessKeyvalOptions*` is unnecessary within the class as it will be managed by the package.

```
⟨*class⟩
\RequirePackage{kvoptions}
\DeclareBoolOption[true]{exetoc}
\DeclareBoolOption[true]{setlist}
\DeclareStringOption[both]{output}
\DeclareStringOption[none]{display}
\DeclareBoolOption[false]{answerspace}
\DeclareStringOption[left]{marginpos}
\DeclareStringOption[expand]{marginwidth}
\DeclareStringOption[left]{noteragged}
\DeclareBoolOption[false]{checkpts}
\DeclareStringOption[false]{correct}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions \relax
\LoadClass{article}
\RequirePackage{exesheet}
\RequirePackage{schooldocs}
⟨/class⟩
```

Options are defined using the kvoptions package. String options are managed through distinct processing macros that are implemented in their respective sections. For options whose effects cannot be dynamically altered and must be configured in the preamble, they are processed once, at `\begin{document}`. The other options are executed when this package is loaded (at the end of the package, as `\exs@process...` commands are not recognized at the outset).

A distinct case is to mention with `setlist` when utilized in conjunction with babel-french. In this instance, this option is processed immediately (further clarification follows below).

```
⟨*package⟩
\@ifclassloaded{exesheet}{}{
    \RequirePackage{kvoptions}
    \DeclareBoolOption[true]{exetoc}
    \DeclareBoolOption[true]{setlist}
    \DeclareStringOption[both]{output}
    \DeclareStringOption[none]{display}
    \DeclareBoolOption[false]{answerspace}
    \DeclareStringOption[left]{marginpos}
    \DeclareStringOption[expand]{marginwidth}
    \DeclareStringOption[left]{noteragged}
    \DeclareBoolOption[false]{checkpts}
    \DeclareStringOption[false]{correct}
}

\ProcessKeyvalOptions*

\PackageInfo{exesheet}{The options ‘notoc’ and ‘nosetlist’
```

```
    \MessageBreak are no longer supported\@gobble}
    % \@gobble suppresses the line number here

\def\exs@process@dynoptions{
    \exs@process@output
    \exs@process@display
    \exs@process@noteragged
} % answerspace do not need a special process macro

\AtEndOfPackage{\exs@process@dynoptions}
\AtBeginDocument{
    \newif\ifexesheet@multicol
    \@ifpackageloaded{multicol}{
        \exesheet@multicoltrue}{\exesheet@multicolfalse}
        % configuring the rule color within answers environments
    \exs@process@setlist
    \exs@process@marginpos
    \exs@process@marginwidth
    \exs@process@checkpts
    \exs@process@correct
    \DisableKeyvalOption[action=warning,package=exesheet]{exesheet}{setlist}
    \DisableKeyvalOption[action=warning,package=exesheet]{exesheet}{marginpos}
    \DisableKeyvalOption[action=warning,package=exesheet]{exesheet}{marginwidth}
    \DisableKeyvalOption[action=warning,package=exesheet]{exesheet}{checkpts}
    \DisableKeyvalOption[action=warning,package=exesheet]{exesheet}{correct}
}
```

`\exesheetset` The `\exesheetset` macro can accept key-val options and can be utilized anywhere in the document to adjust certain settings. However, it won't affect non dynamic options if called outside the preamble. In such cases a warning message occur due to the use of `\DisableKeyValOption`.

```
\def\exesheetset#1{\setkeys{exesheet}{#1}\exs@process@dynoptions}
```

The following old macros (used before version 2.0) provide an alternative to keyval options. They are kept for compatibility reasons.

```
\newcommand{\questionsonly}{
    \PackageWarning{exesheet}{Old command \string\questionsonly\space
        is used. \MessageBreak
        It can be replaced by the option ‘output=questions’}
    \renewcommand\exesheet@output{questions}
    \exs@process@output
}
\newcommand{\answersonly}{
    \PackageWarning{exesheet}{Old command \string\answersonly\space
        is used. \MessageBreak
        It can be replaced by the option ‘output=answers’}
    \renewcommand\exesheet@output{answers}
    \exs@process@output
}
\newcommand{\displaypts}{%
    \PackageWarning{exesheet}{Old command \string\displaypts\space
        is used. \MessageBreak
        It can be replaced by the option ‘display=pts’}
    \renewcommand\exesheet@display{pts}
    \exs@process@display
}
\newcommand{\displaypoints}{%
    \PackageWarning{exesheet}{Old command \string\displaypoints\space
        is used. \MessageBreak
        It can be replaced by the option ‘display=pts’}
    \renewcommand\exesheet@display{pts}
    \exs@process@display
}
\newcommand*{\displaynotes}[1][\RaggedLeft]{%
    \PackageWarning{exesheet}{Old command \string\displaynotes\space
        is used. \MessageBreak
        It can be replaced by the option ‘display=notes’}
    \renewcommand\exesheet@display{notes}
    \exs@process@display
    \renewcommand{\noteragged}{#1}
}
\newcommand*{\displaynotesright}[1][\RaggedRight]{%
    \PackageWarning{exesheet}{Old command \string\displaynotesright
        \space is used. \MessageBreak
        It can be replaced by the options ‘display=notes, margin=right’}
    \renewcommand\exesheet@display{notes}
    \exs@process@display
    \renewcommand\exesheet@margin{right}
    \renewcommand{\noteragged}{#1}
}
```

Now, we load several packages. If the geometry package is already loaded, it will not be reloaded to prevent an option clash. The `shortlabel` option in the enumitem package [3] allows the use of labels similar to the enumerate package such as 1., a), A., and so on. The mparhack package by Tom Sgouros and Stefan Ulrich [9] is loaded exclusively for documents in `twoside` mode.

```
\RequirePackage{ifthen}
\@ifpackageloaded{geometry}{}{\RequirePackage{geometry}}
\RequirePackage{xcolor}
\RequirePackage[shortlabels]{enumitem}
\RequirePackage{tasks}[2020/08/19]
\RequirePackage{versions}
\RequirePackage{fancybox}
\RequirePackage{translations}
\RequirePackage{ragged2e}
\ifthenelse{\boolean{@twoside}}{\RequirePackage{mparhack}}{}
```

7.2 Internationalization

Here we define keywords along with their translations in French, German, Spanish Italian, Portuguese. We achieve this using macros from the translations package by Clemens Niederberger [7]. This package automatically detects the language being used, as loaded by babel or polyglossia.

```
\DeclareTranslationFallback{exesheet-exercise}{Exercise}
\DeclareTranslationFallback{exesheet-subpart}{Part}
\DeclareTranslationFallback{exesheet-annex}{Annex}
\DeclareTranslationFallback{exesheet-ex}{Ex}
\DeclareTranslationFallback{exesheet-points}{points}
\DeclareTranslationFallback{exesheet-point}{point}
\DeclareTranslationFallback{exesheet-correction}{Correction}
\DeclareTranslationFallback{exesheet-pts}{pts}
\DeclareTranslationFallback{exesheet-pt}{pt}

\DeclareTranslation{english}{exesheet-exercise}{Exercise}
\DeclareTranslation{english}{exesheet-subpart}{Part}
\DeclareTranslation{english}{exesheet-annex}{Annex}
\DeclareTranslation{english}{exesheet-ex}{Ex}
\DeclareTranslation{english}{exesheet-points}{points}
\DeclareTranslation{english}{exesheet-point}{point}
\DeclareTranslation{english}{exesheet-correction}{Correction}
\DeclareTranslation{english}{exesheet-pts}{pts}
\DeclareTranslation{english}{exesheet-pt}{pt}

\DeclareTranslation{french}{exesheet-exercise}{Exercice}
\DeclareTranslation{french}{exesheet-subpart}{Partie}
\DeclareTranslation{french}{exesheet-annex}{Annexe}
\DeclareTranslation{french}{exesheet-ex}{Ex}
\DeclareTranslation{french}{exesheet-points}{points}
\DeclareTranslation{french}{exesheet-point}{point}
\DeclareTranslation{french}{exesheet-correction}{Correction}
\DeclareTranslation{french}{exesheet-pts}{pts}
\DeclareTranslation{french}{exesheet-pt}{pt}

\DeclareTranslation{german}{exesheet-exercise}{\"Ubung}
\DeclareTranslation{german}{exesheet-subpart}{Teil}
\DeclareTranslation{german}{exesheet-annex}{Anhang}
\DeclareTranslation{german}{exesheet-ex}{\"Ub}
\DeclareTranslation{german}{exesheet-points}{Punkte}
\DeclareTranslation{german}{exesheet-point}{Punkt}
\DeclareTranslation{german}{exesheet-correction}{Verbesserung}
\DeclareTranslation{german}{exesheet-pts}{P.}
\DeclareTranslation{german}{exesheet-pt}{P.}

\DeclareTranslation{spanish}{exesheet-exercise}{Ejercicio}
\DeclareTranslation{spanish}{exesheet-subpart}{Parte}
\DeclareTranslation{spanish}{exesheet-annex}{Anexo}
\DeclareTranslation{spanish}{exesheet-ex}{Ej}
\DeclareTranslation{spanish}{exesheet-points}{puntos}
\DeclareTranslation{spanish}{exesheet-point}{punto}
\DeclareTranslation{spanish}{exesheet-correction}{Correcci\'on}
\DeclareTranslation{spanish}{exesheet-pts}{ptos}
\DeclareTranslation{spanish}{exesheet-pt}{pto}

\DeclareTranslation{italian}{exesheet-exercise}{Esercizio}
\DeclareTranslation{italian}{exesheet-subpart}{Parte}
\DeclareTranslation{italian}{exesheet-annex}{Annesso}
\DeclareTranslation{italian}{exesheet-ex}{Es}
```

```
\DeclareTranslation{italian}{exesheet-points}{punti}
\DeclareTranslation{italian}{exesheet-point}{punto}
\DeclareTranslation{italian}{exesheet-correction}{Correzione}
\DeclareTranslation{italian}{exesheet-pts}{pti}
\DeclareTranslation{italian}{exesheet-pt}{pt}

\DeclareTranslation{portuges}{exesheet-exercise}{Exerc\'icio}
\DeclareTranslation{portuges}{exesheet-subpart}{Parte}
\DeclareTranslation{portuges}{exesheet-annex}{Anexo}
\DeclareTranslation{portuges}{exesheet-ex}{Ex}
\DeclareTranslation{portuges}{exesheet-points}{pontos}
\DeclareTranslation{portuges}{exesheet-point}{ponto}
\DeclareTranslation{portuges}{exesheet-correction}{Corre\c c\~ao}
\DeclareTranslation{portuges}{exesheet-pts}{pts}
\DeclareTranslation{portuges}{exesheet-pt}{pt}

\newcommand*\exercisename{\GetTranslation{exesheet-exercise}}
\newcommand*\subpartname{\GetTranslation{exesheet-subpart}}
\newcommand*\annexname{\GetTranslation{exesheet-annex}}
\newcommand*\exname{\GetTranslation{exesheet-ex}}
\newcommand*\pointsname{\GetTranslation{exesheet-points}}
\newcommand*\pointname{\GetTranslation{exesheet-point}}
\newcommand*\correctionname{\GetTranslation{exesheet-correction}}
\newcommand*\ptsname{\GetTranslation{exesheet-pts}}
\newcommand*\ptname{\GetTranslation{exesheet-pt}}
```

7.3 Titles

The `exercise` counter assigns numbers to exercises throughout the entire document, regardless of sections. To reset the counter manually, simply use `\setcounter{exercise}{0}`. For an automatic reset at each new section, include the following code in the preamble
`\makeatletter \@addtoreset{exercise}{section} \makeatother`.

The parts counter (`subpart`) depends on the `exercise` counter and is reset with each new exercise.

The commands `\labelexercisestyle` and `\labelsubpartstyle` are initially empty, but they allow you to customize the styling. For example:
`\renewcommand\labelexercisestyle{\sffamily}`.

The `\exe@label` macro, which needs the `exe@check` counter, will be used inside warning messages about the marking scheme (see section 7.6).

By default, the table of contents includes both exercises and parts titles, as controlled by the boolean `\ifexesheet@exetoc`. To only display exercise titles in the table of contents while omitting parts, include the following code in the preamble: `\setcounter{tocdepth}{2}`.

`\exercise`

```
\newcounter{exercise}
\newcounter{exe@check}

\newcommand{\labelexercise}{\exercisename\space \theexercise}
\newcommand{\labelexercisestyle}{}
```

```
\newcommand*{\@exercise}[1][]{%
    \refstepcounter{exercise}
    \subsection*{\labelexercisestyle\labelexercise\enskip #1}
    \ifexesheet@exetoc
        \addcontentsline{toc}{subsection}{\labelexercise}
    \fi
    \ifexesheet@checkpts
     \setcounter{exe@check}{\value{exercise}}
        \def\exe@label{\exercisename\space\theexe@check}
    \fi
}
\newcommand*{\@@exercise}[2][]{%
    \subsection*{\labelexercisestyle #2\enskip #1}
    \setcounter{subpart}{0} % resets the parts counter
    \ifexesheet@exetoc
        \addcontentsline{toc}{subsection}{#2}
    \fi
    \ifexesheet@checkpts \def\exe@label{#2} \fi
}
\newcommand{\exercise}{\@ifstar{\@@exercise}{\@exercise}}
```

\subpart

```
\newcounter{subpart}[exercise] %
\renewcommand{\thesubpart}{\Alph{subpart}}

\newcommand{\labelsubpart}{\subpartname~\thesubpart}
\newcommand{\labelsubpartstyle}{}
\newcommand*{\@subpart}[1][]{%
    \refstepcounter{subpart}%
    \subsubsection*{\labelsubpartstyle\labelsubpart\enskip #1}
    \ifexesheet@exetoc
        \addcontentsline{toc}{subsubsection}{\labelsubpart}
    \fi
}
\newcommand*{\@@subpart}[2][]{%
    \subsubsection*{\labelsubpartstyle #2\enskip #1}
    \ifexesheet@exetoc
        \addcontentsline{toc}{subsubsection}{#2}
    \fi
}
\newcommand{\subpart}{\@ifstar{\@@subpart}{\@subpart}}
```

\annex

```
\newcommand{\annexstyle}{\MakeUppercase}
\newcommand*{\annex}[1][]{%
    \subsection*{\mbox{}\hfill\annexstyle{\annexname} #1\hfill\mbox{}}
    \ifexesheet@exetoc
        \addcontentsline{toc}{subsection}{\annexname}
    \fi
}
```

\exe

```
\newcommand{\exlabel}{\exname.~\theexercise}
\newcommand{\exsepmark}{---}
\newcommand{\@exe}{\bigskip\refstepcounter{exercise}
    \ifexesheet@checkpts
        \setcounter{exe@check}{\value{exercise}}
        \def\exe@label{\exname\space\theexe@check}
    \fi
    \par\noindent\textbf{\exlabel~\exsepmark}~}
\newcommand{\@@exe}{\bigskip\refstepcounter{exercise}
    \ifexesheet@checkpts
        \setcounter{exe@check}{\value{exercise}}
        \def\exe@label{\exname\space\theexe@check}
    \fi
    \par\noindent\textbf{\exlabel}~}
\newcommand{\exe}{\@ifstar{\@@exe}{\@exe}}
```

7.4 Enumerations and lists

\exenumerate The `\setlist` command is part of the enumitem package (`\setenumerate` is deprecated). By default, `itemsep=1ex` is set for first-level lists, and `leftmargin=1.5em` is used to align labels with the start of lines.

```
\newcommand\enumfont{\bfseries}

\newenvironment{exenumerate}[1][]{%
    \setlist[enumerate]{font=\enumfont}
    \setlist[enumerate,1]{leftmargin=1.5em,
        itemsep=3ex plus 1ex minus 1ex,topsep=3ex plus 1ex minus 1ex}
    \setlist[enumerate,3]{noitemsep,nolistsep}
    \setlist[itemize]{noitemsep,nolistsep}
    \begin{enumerate}[#1]
        }{\end{enumerate}}
```

When using the babel-french package, `itemize` lists are altered to use the same dash label for each list level. These modifications are undone here to revert to the default LaTeX `itemize` lists, including labels and spaces. This setting is done by the `\frenchsetup` command, which should be invoked within the `\AtBeginDocument` command or immediately, depending on whether exesheet is loaded before or after babel.

```
\ifexesheet@setlist
    \@ifundefined{frenchsetup}{}{\frenchsetup{StandardLists=true}}
    % must be executed here (and not at begin doc) if loaded after babel
\fi

\newcommand\labelenumone{\arabic{task}.}
\newcommand\labelenuma{(\alph{task})}
\newcommand\refenuma{\alph{task}}

\def\exs@process@setlist{% must be executed at begin document
  \ifexesheet@setlist
    \@ifundefined{frenchsetup}{}{\frenchsetup{StandardLists=true}}
```

```
      % executed at begin doc if loaded before babel
      \setlist[enumerate]{font=\enumfont}
      \setlist[enumerate,1]{topsep=1.5ex plus 1ex minus 1ex,
          leftmargin=1.5em}
    \fi
```

tablenum1 (*env.*) tablenuma (*env.*) The `\NewTasksEnvironment` command is part of the `tasks` package [4]. It enables the definition of the environments `tablenum1`, `tablenuma` and `tablitem`. Horizontal spacing is adjusted to ensure proper alignment with items in other `enumerate` (or `itemize`) environments.

```
    \ifexesheet@setlist
      \settasks{label-format=\enumfont}
      \NewTasksEnvironment[label=\labelenumone,
          column-sep=1em,label-align=right,
          item-indent=1.5em,label-width=1em,label-offset=0.5em,
          after-item-skip=0.5ex plus 0.5ex minus 0.5ex]{tablenum1}[\item](2)
      \NewTasksEnvironment[label=\labelenuma,ref=\refenuma,
          column-sep=1em,label-align=right,
          item-indent=2.15em,label-width=1.6em,label-offset=0.5em,
          after-item-skip=0.5ex plus 0.5ex minus 0.5ex]{tablenuma}[\item](2)
    \else
      \NewTasksEnvironment[label=\labelenumone,
          column-sep=1em,label-align=right,
          label-width=1em,label-offset=0.5em,
          after-item-skip=0.5ex plus 0.5ex minus 0.5ex]{tablenum1}[\item](2)
      \NewTasksEnvironment[label=\labelenuma,ref=\refenuma,
          column-sep=1em,label-align=right,
          item-indent=2.15em,label-width=1.6em,label-offset=0.5em,
          after-item-skip=0.5ex plus 0.5ex minus 0.5ex]{tablenuma}[\item](2)
    \fi
  } % end of macro \exs@process@setlist

  \PackageInfo{exesheet}{The environment 'tablenum' is deprecated
      \MessageBreak and has been replaced by 'tablenum1'\@gobble}
      % \@gobble suppresses the line number here
```

tablitem (*env.*)

```
  \NewTasksEnvironment[label=\labelitemi,
      label-align=right,
      item-indent=2.5em,label-offset=0.5em,
      after-item-skip=0.5ex plus 0.5ex minus 0.5ex]{tablitem}[\item](2)
```

colsenum (*env.*) colsenum* (*env.*) For items aligned by columns, we provide the `colsemnum` and `colsenum*` environments. The `multicol` package is required and an error message is produced if it has not been loaded. `\multicolsep` is the amount of space that should be added above or below the environment.

```
  \newenvironment{colsenum*}[2][]{%
      \ifexesheet@multicol \else
          \PackageError{exesheet}{The environments colsenum and colsenum*
              \MessageBreak need the multicol package}{
              Add \string\usepackage{multicol}\space in the preamble.}
      \fi
      \setlength{\multicolsep}{2ex}
```

```
    \begin{multicols}{#2} % #2 = number of columns
    \begin{enumerate}[#1] % #1 = options of enumerate
    }{
    \end{enumerate}
    \end{multicols}
}

\newenvironment{colsenum}[2][]{%
    \raggedcolumns % default is \flushcolumns
    \begin{colsenum*}[#1]{#2}
    }{
    \end{colsenum*}
}
```

colsitem (*env.*)
colsitem* (*env.*)

The corresponding environments for itemize lists.

```
\newenvironment{colsitem*}[2][]{%
    \ifexesheet@multicol \else
        \PackageError{exesheet}{The environments colsitem and colsitem*
            \MessageBreak need the multicol package}{
            Add \string\usepackage{multicol}\space in the preamble.}
    \fi
    \setlength{\multicolsep}{2ex}
    \begin{multicols}{#2} % #2 = number of columns
    \begin{itemize}[#1] % #1 = options of itemize
    }{
    \end{itemize}
    \end{multicols}
}

\newenvironment{colsitem}[2][]{%
    \raggedcolumns % default is \flushcolumns
    \begin{colsitem*}[#1]{#2}
    }{
    \end{colsitem*}
}
```

7.5 Questions and answers

\exs@process@output

The booleans exesheet@questions and exesheet@answers governs the visibility of their corresponding environments. These booleans are configured through the output key option within the \exs@process@output macro.

```
\newboolean{exesheet@questions}\setboolean{exesheet@questions}{true}
\newboolean{exesheet@answers}\setboolean{exesheet@answers}{true}

\def\exs@process@output{
    \ifthenelse{\equal{\exesheet@output}{questions}}{
        \setboolean{exesheet@questions}{true}
        \setboolean{exesheet@answers}{false}
    }{% else if
    \ifthenelse{\equal{\exesheet@output}{answers}}{
        \setboolean{exesheet@questions}{false}
        \setboolean{exesheet@answers}{true}
        \exesheet@answerspacefalse
    }{% else if
    \ifthenelse{\equal{\exesheet@output}{both}}{
        \setboolean{exesheet@questions}{true}
        \setboolean{exesheet@answers}{true}
        \exesheet@answerspacefalse
    }{% else
    \PackageWarning{exesheet}{Value `\exesheet@output'
         is not supported by `output' option}
    }}}
}
```

`questions` (*env.*) We utilize the versions package developed by Uwe Lück [5], which introduces the macros `\comment` and `\endcomment`. These macros facilitate conditional displays, a technique also employed in the verbatim and version packages. Additionally, the notable codesection package offers the capability to enclose optional code between `\BeginCodeSection{`⟨*skip*⟩`}` and `\EndCodeSection{`⟨*skip*⟩`}` macros, both in the text body and the preamble. However, these macros cannot be used within an environment as we have done here with `\comment` and `\endcomment`. Several of our tests use the LaTeX syntax `\ifthenelse{\bolean{...}}` since `\comment` and `\endcomment` can sometimes interfere with the TeX structure `\if` ...`\else` ...`\fi`.

The two counters `exe@ini` and `subpart@ini` are employed in the subsequent `\set@toclevel` macro.

```
\newcounter{exe@ini}
\newcounter{subpart@ini}

\newenvironment{questions}{
    \ifthenelse{\boolean{exesheet@questions}}{%
        \setcounter{exe@ini}{\value{exercise}}
        \setcounter{subpart@ini}{\value{subpart}}
    }{\comment}
}{\ifthenelse{\boolean{exesheet@questions}}{}{\endcomment}}
```

`answers` (*env.*) The internal macro `\set@toclevel` calculates the title level (counter `toc@level`) to ensure correct typesetting of "Correction" at the start of an `answers` environment, when `questions` and `answers` are displayed together. It involves comparing the `exercise` and `subpart` counters with their values at the time of the `questions` environment call. The `\@enumdepth` counter indicates the current `enumerate` list level (with 0 indicating outside of any list). The optional parameter of the `answers` environment permits the explicit specification of this title level.

```
\newcounter{@toclevel}
\newcommand{\set@toclevel}[1][]{
    \ifthenelse{\equal{#1}{}}{
        \ifthenelse{\(\value{exercise} > \value{exe@ini}\)
           \and \(\value{exe@ini} > 0 \)}{
            \setcounter{@toclevel}{1}
        }{% else
        \ifthenelse{\equal{\the\@enumdepth}{0}}{
            % we're not in an enumerate environment
            \ifthenelse{\(\value{subpart} > \value{subpart@ini}\)
                \or \(\value{subpart} = 0\)}{
              \setcounter{@toclevel}{2}
            }{\setcounter{@toclevel}{3}}
        }{\setcounter{@toclevel}{4}}}
    }{\setcounter{@toclevel}{#1}}}
```

The internal macro `\typeset@correctionname`, displays the term "Correction" at the appropriate level.

```
\definecolor{correctioncolor}{rgb}{0,0.2,0.6} % kind of dark blue
\newcommand{\correctionstyle}{\color{correctioncolor}}

\newcommand{\typeset@correctionname}{
    \ifthenelse{\value{@toclevel} = 1}{
        \section*{\correctionstyle\correctionname}
        \ifexesheet@exetoc
            \addcontentsline{toc}{section}{\correctionname}
        \fi
        \setcounter{exercise}{\value{exe@ini}}
    }{% else if
    \ifthenelse{\value{@toclevel} = 2}{%
        \subsection*{\correctionstyle\correctionname}
        \ifexesheet@exetoc
            \addcontentsline{toc}{subsection}{\correctionname}
        \fi
        \setcounter{subpart}{\value{subpart@ini}}
    }{% else if
    \ifthenelse{\value{@toclevel} = 3}{%
        \subsubsection*{\correctionstyle\correctionname}
        \ifexesheet@exetoc
            \addcontentsline{toc}{subsubsection}{\correctionname}
        \fi
    }{% else
    \par\textbf{\correctionstyle\correctionname}\par
    }}}
}
```

Then we proceed to define the `answers` environment. It seems that the `tasks` package resets the color to black, therefore the `\color{correctioncolor}` options in `\settasks`.

```
\newenvironment{answers}[1][]{% #1 is the optional level
    \ifthenelse{\boolean{exesheet@answers}}{%
        \ifthenelse{\boolean{exesheet@questions}}{%
            \set@toclevel[#1]%
            \typeset@correctionname%
            \correctionstyle%
            \ifexesheet@setlist
                \settasks{
                    label-format = \color{correctioncolor}\enumfont,
                    item-format  = \color{correctioncolor}
                }%
            \else
                \settasks{
                    label-format = \color{correctioncolor},
                    item-format  = \color{correctioncolor}
                }%
            \fi%
            \ifexesheet@multicol
                \renewcommand{\columnseprulecolor}{%
                    \color{correctioncolor}}
            \fi%
        }{}%
    }{\comment}
}{\ifthenelse{\boolean{exesheet@answers}}{%
    \setcounter{exe@ini}{0}
    \setcounter{subpart@ini}{0}
    }{\endcomment}}

\newenvironment{answers*}{
    \ifthenelse{\boolean{exesheet@answers}}{\correctionstyle}{\comment}
}{\ifthenelse{\boolean{exesheet@answers}}{}{\endcomment}}
```

When placing `\correctionstyle` before `\subsubsection` in the `answers` environment (as in the case of `\typeset@correctionname`), the preceding vertical space may become too wide.

`\question`
`\question*`

```
\newcommand{\@question}[1]{\ifexesheet@questions #1\fi}
\newcommand{\@@question}[1]{%
    \ifexesheet@questions\ifexesheet@answers \else #1\fi\fi}
\newcommand{\question}{\@ifstar{\@@question}{\@question}}
```

`\answer`
`\answer*`

```
\newcommand{\@answer}[1]{%
    \ifexesheet@answers%
        \ifexesheet@questions {\correctionstyle #1}\else #1\fi
    \fi
}
\newcommand{\@@answer}[1]{%
    \ifexesheet@answers\ifexesheet@questions \else #1\fi\fi}
\newcommand{\answer}{\@ifstar{\@@answer}{\@answer}}
```

`\answerspace` The `\answerspace` macro leaves blank space to allow students for writing their answers on the provided paper following a suggestion by Maxime Chupin.

```
\newcommand\answerspace[1]{
    \ifexesheet@answerspace \par\vspace{#1} \fi}
```

`\exs@process@correct` The `correct` option needs the schooldocs package. It triggers the `\correct` macro of schooldocs which adds the content of `\correctname` in the title of the document. Here the option `conditional` triggers `\correct` only if `output=answers` or `both`.

```
\def\exs@process@correct{
    \ifthenelse{\equal{\exesheet@correct}{false}}{% do nothing
    }{% else
    \@ifpackageloaded{schooldocs}{
        \ifthenelse{\equal{\exesheet@correct}{true}}{
            \correct
        }{% else
        \ifthenelse{\equal{\exesheet@correct}{conditional}}{
            \ifexesheet@answers \correct \fi
        }{}}
    }{
        \PackageWarningNoLine{exesheet}{The ‘correct’ option requires
            \MessageBreak
            the ‘schooldocs’ package to be loaded}
    }}
}
```

7.6 Marking scheme options processing

The options `display`, `marginpos`, `marginwidth` and `noteragged` are handled using the following internal commands.

The `display` key option determines the value of the booleans `exesheet@pts` and `exesheet@notes`. The `exesheet@pts` boolean controls the display of the content of `\pts` and optional arguments of `\note`, while the `exesheet@notes` boolean controls mandatory arguments of `\note`.

`\exs@process@display`

```
\newboolean{exesheet@pts}
\newboolean{exesheet@notes}

\def\exs@process@display{
    \ifthenelse{\equal{\exesheet@display}{pts}}{
        \setboolean{exesheet@pts}{true}
        \setboolean{exesheet@notes}{false}
    }{% else if
    \ifthenelse{\equal{\exesheet@display}{notes}}{
        \setboolean{exesheet@pts}{true}
        \setboolean{exesheet@notes}{true}
    }{% else if
    \ifthenelse{\equal{\exesheet@display}{none}}{
        \setboolean{exesheet@pts}{false}
        \setboolean{exesheet@notes}{false}
    }{% else
    \PackageWarning{exesheet}{Value ‘\exesheet@display’
         is not supported by ‘display’ option}
    }}}
}
```

`\exs@process@marginpos` The `marginpos` key option takes the values `left` (the default value) or `right` (or `inner` and `outer`). In practice, `inner` is equivalent to `left`, but in two-sided mode, the values `left` or `right` are converted to `outer` (which is then the default value for two-sided mode).

```
\newboolean{exesheet@leftmargin}
```

```
\def\exs@process@marginpos{
    \ifthenelse{\equal{\exesheet@marginpos}{left}}{
        \if@twoside%
            \PackageWarningNoLine{exesheet}{The default ‘marginpos’
                option \MessageBreak
                for two-sided documents is ‘outer’.\MessageBreak
                To change the side, use ‘inner’}
            \def\exesheet@marginpos{outer}
            \setboolean{exesheet@leftmargin}{false}
            \normalmarginpar
        \else% default
            \setboolean{exesheet@leftmargin}{true}
            \reversemarginpar
        \fi
    }{% else if
    \ifthenelse{\equal{\exesheet@marginpos}{right}}{
        \if@twoside%
            \PackageWarningNoLine{exesheet}{The default ‘marginpos’
                option \MessageBreak
                for two-sided documents is ‘outer’.\MessageBreak
                To change the side, use ‘inner’}
            \def\exesheet@marginpos{outer}
        \fi
        \setboolean{exesheet@leftmargin}{false}
        \normalmarginpar
    }{% else if
    \ifthenelse{\equal{\exesheet@marginpos}{inner}}{
        \setboolean{exesheet@leftmargin}{true}
        \reversemarginpar
    }{% else if
    \ifthenelse{\equal{\exesheet@marginpos}{outer}}{
        \setboolean{exesheet@leftmargin}{false}
        \normalmarginpar
    }{% else
    \PackageWarningNoLine{exesheet}{The value ‘\exesheet@marginpos’
         is not supported by the ‘marginpos’ option}
    }}}}
}
```

`\exs@process@marginwidth` The `marginwidth` option adjusts the ratio between left and right margins based on what needs to be displayed in the margin (points only or full notes)[10].

When `display=notes`, the additional length of `1 in` corresponds to the default free space to the left of `\oddsidemargin`.

The macros `\standardmarginwidthfactor` and `\largemarginwidthfactor` represent the ratios between the total margin width and `\marginparwidth`.

```
\def\standardmarginwidthfactor{0.6}
\def\largemarginwidthfactor{0.8}

\newcommand*{\leftnotemarginwidth}[1]{
```

[10]To ensure the accurate effect on the margin ratio, this option is processed at the beginning of the document, after other commands that could potentially alter the page geometry.

```
    \setlength{\marginparwidth}{\oddsidemargin}
    \addtolength{\marginparwidth}{1in}
    \addtolength{\marginparwidth}{-\marginparsep}
    \setlength{\marginparwidth}{#1\marginparwidth}
}

\newcommand*\rightnotemarginwidth[1]{
    \setlength{\marginparwidth}{\paperwidth}
    \addtolength{\marginparwidth}{-\textwidth}
    \addtolength{\marginparwidth}{-\oddsidemargin}
    \addtolength{\marginparwidth}{-\marginparsep}
    \addtolength{\marginparwidth}{-1in}
    \setlength{\marginparwidth}{#1\marginparwidth}
}

\def\exesheet@smallmargins{
    \geometry{hmarginratio=1:1}
    \leftnotemarginwidth{\standardmarginwidthfactor}
}
\def\exesheet@standardmargins{
    \ifexesheet@leftmargin
        \geometry{hmarginratio=3:2}
        \leftnotemarginwidth{\standardmarginwidthfactor}
    \else
        \geometry{hmarginratio=2:3}
        \rightnotemarginwidth{\standardmarginwidthfactor}
    \fi
}
\def\exesheet@largemargins{
    \ifexesheet@leftmargin
        \geometry{hmarginratio=3:1}
        \leftnotemarginwidth{\largemarginwidthfactor}
    \else
        \geometry{hmarginratio=1:3}
        \rightnotemarginwidth{\largemarginwidthfactor}
    \fi
}

\def\exs@process@marginwidth{
    \ifthenelse{\equal{\exesheet@marginwidth}{standard}}{
        \ifthenelse{\equal{\exesheet@display}{none}}{
            \if@twoside
                \exesheet@standardmargins
            \else
                \exesheet@smallmargins
            \fi
        }{% else display=pts or display=notes
            \exesheet@standardmargins
        }
    }{% else if
    \ifthenelse{\equal{\exesheet@marginwidth}{expand}}{
        \ifthenelse{\equal{\exesheet@display}{none}}{
            \if@twoside
                \exesheet@standardmargins
            \else
                \exesheet@smallmargins
            \fi
        }{% else if
        \ifthenelse{\equal{\exesheet@display}{pts}}{
            \exesheet@standardmargins
        }{% else display=notes
            \exesheet@largemargins
        }}
    }{% else if
        \ifthenelse{\equal{\exesheet@marginwidth}{unset}}{
        % do nothing
    }{% else
    \PackageWarningNoLine{exesheet}{The value ‘\exesheet@marginwidth’
         is not supported by the ‘marginwidth’ option}
    }}}
}
```

For a two-sided document, the geometry package does not correctly set the default width of the margin paragraph; it's too wide. Therefore, we provide an explicit setting here, which is useful when `marginwidth=unset`. Otherwise, the setting is handled by the `marginwidth` key option.

```
\if@twoside \rightnotemarginwidth{0.5} \fi
```

`\exs@process@noteragged` The noteragged option can take one of the following values: `left`, `right`, `center`, `justify` or `twoside`. When working with a two-sided document, `\marginpar` can be used with an optional parameter to distinguish left from right contents. In this context, we employ `\noteraggedleft` and `\noteraggedright` instead of `\noteragged`. The ragged2e package by Martin Schröder [8] offers the commands `\RaggedLeft`, `\RaggedRight`, `\Centering`, and `\justifying`. These commands yield better results compared to the standard `\raggedleft`, `\raggedright` and `\centering` commands. Margin paragraphs are justified by default in LaTeX.

```
\newcommand{\noteragged}{}
\newcommand{\noteraggedleft}{}
\newcommand{\noteraggedright}{}

\def\exs@process@noteragged{
    \ifthenelse{\equal{\exesheet@noteragged}{left}}{
        \if@twoside
            \renewcommand{\noteraggedleft}{\RaggedLeft}
            \renewcommand{\noteraggedright}{\RaggedLeft}
        \else
            \renewcommand{\noteragged}{\RaggedLeft}
        \fi
    }{% else if
    \ifthenelse{\equal{\exesheet@noteragged}{right}}{
        \if@twoside
            \renewcommand{\noteraggedleft}{\RaggedRight}
            \renewcommand{\noteraggedright}{\RaggedRight}
        \else
            \renewcommand{\noteragged}{\RaggedRight}
        \fi
    }{% else if
    \ifthenelse{\equal{\exesheet@noteragged}{center}}{
        \if@twoside
            \renewcommand{\noteraggedleft}{\Centering}
            \renewcommand{\noteraggedright}{\Centering}
        \else
            \renewcommand{\noteragged}{\Centering}
        \fi
    }{% else if
    \ifthenelse{\equal{\exesheet@noteragged}{justify}}{
          \renewcommand{\noteraggedleft}{\justifying} % equiv to nothing
          \renewcommand{\noteraggedright}{\justifying}
          \renewcommand{\noteragged}{\justifying}
    % justify is the default LaTeX setting
    }{% else  if
    \ifthenelse{\equal{\exesheet@noteragged}{twoside}}{
        \if@twoside
            \renewcommand{\noteraggedleft}{\RaggedLeft}
            \renewcommand{\noteraggedright}{\RaggedRight}
        \else
            \PackageWarning{exesheet}{Invalid option 'noteragged=twoside'
             when the document \MessageBreak is not in two-side mode}
        \fi
    }{% else
    \PackageWarning{exesheet}{The value '\exesheet@noteragged'
         is not supported by the 'noteragged' option}
    }}}}}
}
```

`\exs@process@checkpts` The scale control option relies on calculations with *lengths*, which need to have a *global* scope.

For questions, assigned points will be added in `\sum@pts`, while for exercises, points accumulate in `\sum@exe`. These lengths are compared against `\exe@total` and `\sheet@total`. The `\exe@check` macro validates the calculations of the previous exercise when triggered by `\points`, `\totalexe` or `\totalpoints` macros. Percent symbols at end of lines are necessary to prevent unwanted spaces. `\exe@check` is also invoked within `\exs@process@checkpts` at the document's end for a final check on the last exercise.

```
\newlength{\sheet@total}
\newlength{\sum@exe}
\newlength{\exe@total}
\newlength{\sum@pts}
\def\exe@currentlabel{none}
\newboolean{scale@valid}

\def\exe@check{%
    \ifthenelse{\lengthtest{\sum@pts = 0pt}}{%
    % do not check, no points or first exercise begins
    \ifthenelse{\equal{\exe@currentlabel}{none}}{}{%
        \PackageWarningNoLine{exesheet}{\exe@currentlabel:
            \the\exe@total}}%
    }{%
        \ifthenelse{\lengthtest{\exe@total = \sum@pts}}{%
            \PackageWarningNoLine{exesheet}{\exe@currentlabel:
                Sum of points \the\exe@total\space is valid}%
        }{%
        \PackageWarningNoLine{exesheet}{\exe@currentlabel:
            Sum of points is \the\sum@pts\space
            instead of \the\exe@total}%
        \setboolean{scale@valid}{false}%
        }%
    }%
}

\def\exs@process@checkpts{
    \ifexesheet@checkpts
        \ifthenelse{\lengthtest{\sheet@total = 0pt}}{
            \PackageWarningNoLine{exesheet}{Option checkpts is true,
                \MessageBreak
                but \string\totalsheet\space is missing
                in the preamble. \MessageBreak
                See documentation}
        }{}
        \global\sum@exe=0pt
        \global\exe@total=0pt
        \global\sum@pts=0pt
        \setboolean{scale@valid}{true}
        \AtEndDocument{% final checking (global)
            \ifthenelse{\equal{\exe@currentlabel}{none}}{
              \ifthenelse{\lengthtest{\sum@pts = 0pt}}{
                \PackageWarningNoLine{exesheet}{checkpts:
                    No points displayed}
              }{
                \ifthenelse{\lengthtest{\sheet@total = \sum@pts}}{
                    \PackageWarningNoLine{exesheet}{Total:
                        Sum of points \the\sheet@total\space is valid}
                }{
                    \PackageWarningNoLine{exesheet}{Total:
                        Sum of points is \the\sum@pts\space
                        instead of \the\sheet@total}
                }}
            }{% last exercise and final checking
              \exe@check
              \ifthenelse{\lengthtest{\sum@exe} = 0pt}{
                \PackageWarningNoLine{exesheet}{checkpts:
                    No points displayed}
                }{
                \ifthenelse{\lengthtest{\sheet@total = \sum@exe}}{
                    \PackageWarningNoLine{exesheet}{Total:
                        Sum of points \the\sheet@total\space is valid}
                }{
                    \PackageWarningNoLine{exesheet}{Total:
                        Sum of points is \the\sum@exe\space
                        instead of \the\sheet@total}
                    \setboolean{scale@valid}{false}
                }
                \ifthenelse{\boolean{scale@valid}}{
                    \PackageWarningNoLine{exesheet}{
                        Marking scheme checked without errors}
                }{
                    \PackageWarningNoLine{exesheet}{
                        Marking scheme checked with ERRORS! See above}
                }
              }
            }
        }
    \fi
}
```

7.7 Marking scheme commands

The `\check@points` macro, used by `\points` and `\totalexe`, triggers the marking scheme control (with `\exe@check` defined above) and sets label and lengths for the next exercise.

```
\newcommand*{\check@points}[1]{%
    \ifexesheet@checkpts%
        \exe@check% checks the previous exercise
        \gdef\exe@currentlabel{\exe@label}% for the upcoming exercise
        \global\sum@pts=0pt%
        \global\exe@total=#1pt%
        \global\advance\sum@exe by #1pt%
    \fi%
}
```

`\points`

```
\definecolor{pointscolor}{named}{red}
\newcommand{\pointsstyle}{%
    \small\mdseries\sffamily\color{pointscolor}\fbox}
\newcommand*{\points}[1]{%
    \ifthenelse{\boolean{exesheet@questions}}{\hfill
        \pointsstyle{#1~%
            \ifthenelse{\lengthtest{#1pt < 2pt}}{\pointname}{\pointsname}}%
        \check@points{#1}%
    }{}
}
```

To prevent spaces between the `\fbox` and its inner text, percent symbols are necessary. The test `#1 < 2` doesn't work with decimal numbers without `\lengthtest`, but it works with lengths.

`\pts`

```
\definecolor{ptscolor}{named}{red}
\newcommand{\ptsstyle}[1]{%
    \footnotesize\centering\sffamily\color{ptscolor} (#1)}
\newcommand*{\ptsmark}[1]{%
```

```
    \ifthenelse{\lengthtest{#1pt < 2pt}}{#1 \ptname}{#1 \ptsname}}
\newcommand*{\pts}[1]{%
    \ifexesheet@pts%
        \mbox{}%
        \marginpar{\hspace{0pt}\ptsstyle{\ptsmark{#1}}}%
        \ifexesheet@checkpts%
            \global\advance\sum@pts by #1pt%
        \fi%
    \fi%
    \ignorespaces
}
```

`\totalexe` In the subsequent macros that utilize `\marginpar`, the presence of percent symbols and `\ignorespaces` is essential to prevent the occurrence of expanded blank spaces in the text (or the margin), where these macros are incorporated.

```
\definecolor{markingcolor}{named}{red}
\newcommand{\markingstyle}[1]{\footnotesize\sffamily%
    \centering\color{markingcolor}\textbf{#1}}
    % inner arguments enable the implementation of boxed styles
\newlength{\ptsboxlength}
\setlength{\ptsboxlength}{3.1em}
\cornersize{1}
\newcommand*{\totalexe}[1]{%
    \ifexesheet@pts%
        \mbox{}%
        \marginpar{\hspace{0pt}\markingstyle{\ovalbox{%
            \makebox[\ptsboxlength]{\ptsmark{#1}}}}}%
        \check@points{#1}%
    \fi%
    \ignorespaces
}
```

`\totalsheet`

```
\newcommand*{\totalsheet}[1]{
    \global\sheet@total=#1pt
}
```

`\note` `\note*` The booleans `exesheet@pts` and `exesheet@notes` control the display of marginal notes. If `exesheet@pts` is set to `false`, `exesheet@notes` will be ignored. `\noindent` is required when using `\justifying` from the `ragged2e` package [8]. Within the `\note@marginpar` macro, enclosing `\markingstyle` in double braces helps prevent unintended formatting within the mandatory argument of `\note`. A vicious error occurs when using an `\if` ...`\fi` structure instead of `\ifthenelse` inside `\note@marginpar` (but only if `@twoside` is `true`).

```
\definecolor{notecolor}{rgb}{0.0, 0.4, 0.0} % kind of dark green
\newcommand{\notestyle}[1]{\footnotesize\sffamily\color{notecolor} #1}
\newcommand{\note@marginpar}[1]{%
    \if@twoside%
        \marginpar[\noteraggedleft #1]{\noteraggedright #1}%
    \else%
```

```
        \marginpar{\noteragged #1}%
    \fi%
}
\newcommand{\@note}[2][]{%
    \ifexesheet@pts%
        \mbox{}%
        \note@marginpar{%
            \ifthenelse{\equal{#1}{}}{}{{%
                \noindent\hspace{0pt}\markingstyle{#1}\\}}%
            \ifthenelse{\boolean{exesheet@notes}}{%
                \noindent\hspace{0pt}\notestyle #2%
            }{}%
        }%
        \ifexesheet@checkpts%
            \ifthenelse{\equal{#1}{}}{}{%
                \global\advance\sum@pts by #1pt%
            }%
        \fi%
    \fi%
    \ignorespaces
}
\newcommand{\@@note}[1]{%
    \ifexesheet@pts%
        \mbox{}%
        \marginpar{\noindent\hspace{0pt}\markingstyle{#1}}%
        \ifexesheet@checkpts%
            \global\advance\sum@pts by #1pt%
        \fi%
    \fi%
    \ignorespaces
}
\newcommand{\note}{\@ifstar{\@@note}{\@note}}
```

`\totalpoints`

```
\newcommand{\totalpoints}{%
    \ifthenelse{\boolean{exesheet@pts}}{\totalexe}{\points}}
```

⟨/package⟩

References

[1] *The schooldocs package*, Antoine Missier, CTAN, v1.4 2023/12/28.

[2] *The geometry package*, Hideo Umeki, CTAN, v5.9 2020/01/02.

[3] *Customizing lists with the enumitem package*, Javier Bezos, CTAN, v3.9 2019/06/20.

[4] *tasks – lists with columns filled horizontally*, Clemens Niederberger. CTAN, v1.4a, 2022/01/08.

[5] *The versions package – Omit passages optionally under LaTeX*, Uwe Lück, CTAN, v0.55 2005/04/28.

[6] *Documentation for `fancybox.sty`: Box tips and tricks for LaTeX*, Timothy Van Zandt, CTAN, v1.4 2010/05/15.

[7] *translations – Internationalization of LaTeX 2ε Packages*, Clemens Niederberger, CTAN, v1.12 2022/02/05.

[8] *The ragged2e-package*, Martin Schröder, CTAN, v3.6 2023/06/22.

[9] *`mparhack.sty`*, Tom Sgouros, Stefan Ulrich, CTAN, v1.5 2021/05/02.

[10] *The sectsty package*, Rowland McDonnell, CTAN, v2.0.2 2002/02/25.

[11] *`exercise.sty` : a package to typeset exercises*, Paul Pichaureau, CTAN, v1.6 2014/10/21.

[12] *The exercises package*, Roger Jud, CTAN, v1.1 2000/05/17.

[13] *xsim – eXercise Sheets IMproved – the official successor of the exsheets package*, Clemens Niederberger, CTAN, v0.21 2022/02/12.

[14] *The exframe package*, Niklas Beisert, CTAN, v3.4 2020/02/24.

[15] *Using the exam document class*, Philip Hirschhorn, CTAN, v2.704 2023/07/09.

[16] *answers Production of solution sheets in LaTeX 2ε*, Mike Piff and Joseph Wright, CTAN, 2.16 2014/08/24.

[17] *The probsoln v3.05: creating problem sheets optionally with solutions*, Nicola L.C. Talbot, CTAN 2017/07/10.

[18] *The ExSol package*, Walter Daems, CTAN, 1.4 2018/10/23.

[19] *The exercisepoints Package*, Henning Kerstan, CTAN, v1.2.3 2019/01/03.

[20] *worksheet*, Benjamin Zöllner, CTAN, v1.1 2018/08/17.

[21] *exam-n: exam papers*, Norman Gray, CTAN, v1.4.0 2022/10/10.

[22] *The eqexam Package – part of the AcroTeX eDucation Bundle*, D. P. Story, CTAN, 5.2 2021/02/26.

[23] *cesenaexam — class file to typeset exams*, Alex Pacini, CTAN, v2.0 2017/08/03.

[24] *Package esami*, Grazia Messineo, Salvatore Vassallo, CTAN, v2.8 2023/07/21.

[25] *Teh randexam class for LaTeX*, Jianrui Lyu, CTAN, 2024D, 2024/02/03.

[26] *The hideanswer package: generate documents with and without answers by toggling a switch*, Yukoh Kusakabe, CTAN, v1.1 2022/07/09.

[27] *The mathexam Package*, Jan Hlavacek, CTAN, v1.00 2007/07/30.