THE EXSHEETS BUNDLE

v0.9c 2013/04/21

the packages EXSHEETS, TASKS, TRANSLATIONS and CNTFORMATS
or
Yet another package for the creation of exercise sheets and exams.

Clemens NIEDERBERGER

https://bitbucket.org/cgnieder/exsheets/
contact@mychemistry.eu

English documentation

EXSHEETS provides means to create exercises or questions and their corresponding solutions. The questions can be divided into classes and can be printed selectively. Meta-data to questions can be added and recovered. The solutions may be printed where they are, can be collected and printed at a later point in the document alltogether or section-wise or selectively by ID. EXSHEETS provides a comprehensive interface for styling the headings of questions and solutions.

Contents

Part I
Preliminary

1 Licence and Requirements

ExSheets is placed under the terms of the LaTeX Project Public License, version 1.3 or later (`http://www.latex-project.org/lppl.txt`). It has the status "maintained."

ExSheets loads and needs the following packages: l3kernel, xparse,[1] xtemplate,[2] l3sort,[3] l3keys2e,[4] xcolor,[5] ulem,[6] etoolbox,[7] environ,[8] and pgfcore.[9] ExSheets calls `\normalem` (from the ulem package).

[1] CTAN: xparse [2] CTAN: xtemplate [3] CTAN: l3sort [4] CTAN: l3keys2e [5] CTAN: xcolor [6] CTAN: ulem
[7] CTAN: etoolbox [8] CTAN: environ [9] CTAN: pgfcore

2 Motivation

There are already quite a number of packages that allow the creation of exercise sheets or written exams. Just to name the most common ones: exam[10] [4], examdesign[11] [1], exercise[12] [12], probsoln[13] [14], answers[14] [13] (and many more).

One thing I missed in all packages that I've tried out[15] was a high flexibility in choosing which questions and solutions should be printed, where which solutions should be printed and so on, combined with the possibility to assign questions to different classes so one could for example create two versions of an exam out of the box. And – I can't get enough – I also want to be able to use/design different layouts for questions additional to a standard section-like format. All these points are realized in ExSheets.

Additionally one should be able to assign some sort of meta-data to questions that of course should be easily reusable. How this can be done is explained in section 12.

Then there is – at least in Germany – the habit of having lists of exercises aligned in columns but counting from the left to the right instead from up to down. ExSheets provides a possibility for that (see part III). I am not quite content with it as it works now, though.[16]

On the other hand ExSheets doesn't – and probably won't – offer a real possibility for creating multiple choice questions. As a fact it doesn't provide many (if any) means to specify the *type* of question or the structure. If these are your needs take a look at examdesign, for example. Or exploit the possibilities enumitem[17] [2] gives you.

I had the idea for this package in 2008. Back then my TEX skills were by far not good enough to write it. Actually, even today I wouldn't have been able to realize it without all the l3 packages like l3kernel and l3packages. I actively began to develop ExSheets in spring 2011 but it wasn't until now (September 2012) that I consider it stable enough for wider usage. At the time of writing (April 21, 2013) there still are probably lots of rough edges let alone bugs so I am very interested in all kinds of feedback.

3 Additional Packages

ExSheets actually bundles four packages: ExSheets, tasks, translations and cntformats. tasks is described in part III, translations is described in part IV and cntformats in part V. These packages provide functionality that is used by ExSheets. They can, however, be used independently from ExSheets.

The packages translations and cntformats both aren't really useful on a user-level but maybe for package writers.

4 Installation

If you install ExSheets manually beware to put the files

[10] CTAN: exam [11] CTAN: examdesign [12] CTAN: exercise [13] CTAN: probsoln [14] CTAN: answers [15] Well, probably I didn't try hard enough… [16] There are still other possibilities, for example take a look here: `http://tex.stackexchange.com/questions/67966/enumerate-in-multicols` or at the multienum package [5]. [17] CTAN: enumitem

```
exsheets_headings.def

exsheets_headings.cfg
```

in the same directory as the `exsheets.sty` file.[18] You *can* install the other packages, TASKS, TRANSLATIONS and CNTFORMATS, in different locations but since they belong to ExSheets they probably should be placed in the same directory.

You should put the file `tasks.cfg` in the same directory as the `tasks.sty` file.

As with every manual package installation you need to make sure to put the files in a directory where TEX can find them and afterwards update the database.

5 News

With version 0.7 there has been a potentially breaking change: the `tasks` environment previously provided by ExSheets has been extracted into a package of its own. This does not change any syntax *per se*. However, if you used custom settings then you'll probably run into some problems. The options for the environment are no longer set with `\SetupExSheets` but with `\settasks`. Also the object that is used for the list template and its instances has been renamed from `exsheets-tasks` into `tasks`.

What's probably even more of a breaking change is a syntax difference of the `tasks` environment: the optional argument for the number of columns is *no longer set in braces but parentheses*. This is deliberate as it reflects the optional nature of the argument better and is consistent with the syntax of `\NewTasks`, too.

Additionally the labels of the list got an additional offset of `1ex` from the items which will lead to slightly different output. In some cases this might actually lead to the most annoying changes. In this case say `\settasks{label-offset=0pt}` which should cure things again.

I am very sorry for any inconvenience! I am trying to keep such changes as minimal and rare as possibly. However, it is not always avoidable when a package is new and still a child. It will grow up eventually.

Part II
The ExSheets package

6 Setup

The ExSheets package has three different types of options, kind of. The first type are the classic package options which are used when you load ExSheets:

[18] That is, a directory like `texmf-local/tex/latex/exsheets`, probably

```
\usepackage[<options>]{exsheets}
```

These options are described in section 7.

The second type are options that belong to a specific environment or command. These options are either used directly with the environment/command

```
\begin{env}[<options>]
 ...
\end{env}
```

or can be set with the setup command:

▶ `\SetupExSheets[<module>]{<options>}`

The options of the second type all belong to `modules`. Let's say you want to specify some options of the `question` environment. You can then say the following:

```
\SetupExSheets[question]{option1,option2=value2}
% or:
\SetupExSheets{question/option1,question/option2=value2}
```

The `module` an option belongs to is written in the left margin next to the when the option is described.

The third type aren't options at all, actually. However, thanks to the great xtemplate package you are able to define your own instances of some of the objects used by ExSheets. This is explained in a little more detail in part 18 on page 32 ff.

7 Package Options

The package ExSheets has some options, namely the following ones:

▶ `color = <colour>` Default: `exsheetsblue`

A custom colour that is used in certain but very rare circumstances.

▶ `counter-format = <counter-format>` Default: `qu.`

Formatting of the counter of the questions. This option takes a special kind of string that is described in section 8.5.

▶ `counter-within = <counter>` (initially empty)

Resets the `question` counter with every step of `<counter>`.

▶ `auto-label = true|false` Default: `false`
If set to true ExSheets will automatically place a `\label{qu:<id>}` for each question. It will also create the question properties `ref` and `pageref`, see section 12 for more on this.

▶ `headings = <instance>` Default: `block`
Choose the style of the questions' and solutions' headings. There are two predefined styles: `block` and `runin`.

▶ `headings-format = <code>` Default: `\normalsize\bfseries`
This code is placed immediately before the headings of the questions and solutions.

▶ `load-headings = true|false` Default: `false`
Loads additional styles for the headings. More on this is described in section 18.2.

▶ `load-tasks = true|false` Default: `false`
Loads additional styles for the `tasks` environment. See part III.

▶ `totoc = true|false` Default: `false`
This option adds the questions and solutions with their names and numbers to the table of contents.

▶ `questions-totoc = true|false` Default: `false`
This option adds the questions with their names and numbers to the table of contents.

▶ `solutions-totoc = true|false` Default: `false`
This option adds the solutions with their names and numbers to the table of contents.

▶ `toc-level = <toc level>` Default: `subsection`
This option sets the level in which questions and solutions should appear in the table of contents.

▶ `questions-toc-level = <toc level>` Default: `subsection`
This option sets the level in which questions should appear in the table of contents.

▶ `solutions-toc-level = <toc level>` Default: `subsection`
This option sets the level in which solutions should appear in the table of contents.

▶ `use-ref = true|false` Default: `false`
enable referencing to sections and chapters in a way that the references can be used with `\printsolutions`, see section 10.2 for details.

The `toc` options are demonstrated with section VI and the solutions printed there being listed in the table of contents.

8 Create Questions/Exercises and their Solutions

Now, let's start with the most important part: the questions and (possibly) their respective solutions.

8.1 The question Environment

Questions are written inside the question environment:

► \begin{question}[<options>]{<points>} … \end{question}

```
\begin{question}
 This is our very first very difficult to solve question!
\end{question}
```

Exercise 1.
This is our very first very difficult to solve question!

As you can see a heading is automatically created and the question is numbered. You can of course change both the numbering and the naming, but more on that later.

The question environment takes an optional argument {<points>} that can be used to assign points to the question (as is common in written exams):

```
\begin{question}{3}
 This is our first difficult question that is worth 3 points!
\end{question}
```

Exercise 2. 3 P.
This is our first difficult question that is worth 3 points!

These points are saved internally (see section 9 for reasons why) and are written to the right margin next to the question heading in the default setting.

You can also assign bonus points by inserting <point>+<bonus points> as argument.

```
\begin{question}{1+1}
 This question is worth 1 point and 1 bonus point.
\end{question}
\begin{question}{+3}
 This question is a bonus question. It is worth 3 bonus points.
\end{question}
```

Exercise 3. 1 (+1) P.
This question is worth 1 point and 1 bonus point.

Exercise 4. (+3 P.)
This question is a bonus question. It is worth 3 bonus points.

Introduced in version 0.3

On additional thing: you might want to define custom commands that should behave differently if they're inside or outside of the question environment. In this case you can use these commands:

▷ `\IfInsideQuestionTF{<true code>}{<false code>}`

▷ `\IfInsideQuestionT{<true code>}`

▷ `\IfInsideQuestionF{<false code>}`

8.2 Options to the question Environment

The `question` environment takes one or more of the following options:

`question` ▶ `type = exam|exercise` Default: `exercise`
determines the type of question and changes the default name of a question from "Exercise" to "Question". These default names are language dependent.
If you use `\usepackage[ngerman]{babel}`, for example, then the names are "Übung and "Aufgabe".

`question` ▶ `name = <name>` (initially empty)
sets a custom name. All predefined names are discarded.

`question` ▶ `print = true|false` Default: `true`
prints or hides the question.

`question` ▶ `ID = <id>` (initially empty)
assigns a custom ID to the question. See section 10.3 for further information.

`question` ▶ `label = <label>` (initially empty)
Places a `\label{<label>}` for the question. This will overwrite any label that is placed by the `auto-label` option.

`question` ▶ `class = <class>` (initially empty)
assigns a class to the question. See section 11.1 for further information.

`question` ▶ `topic = <topic>` (initially empty)
assigns a topic to the question. See section 11.2 for further information.

`question` ▶ `use = true|false` Default: `true`
discards the question. Or not.

```
\begin{question}[type=exam]
 This question has the type \texttt{exam}. The default name has changed
  from
 ''Exercise'' to ''Question''.
\end{question}
\begin{question}[name=Fancy name]
 This question has a custom name.
\end{question}
\begin{question}[print=false]
 This question is not printed.
\end{question}
```

Question 5.
This question has the type `exam`. The default name has changed from “Exercise” to “Question”.

Fancy name 6.
This question has a custom name.

The difference between `print` and `use` lies behind the scenes: with `print = false` the question is not printed, but it still gets an individual ID, is numbered, and a possible solution is saved. This is for example useful when you want to print a sample solution for an exam. With `use = false` it is fully discarded which means it is not accessible through an ID and a possible solution will not be saved.

8.3 The `solution` Environment

If you want to save/print (more on the exact usage in section 10) a solution you have to use the `solution` environment *after* the question it belongs to and *before* the next question.

► `\begin{solution}[<options>]` … `\end{solution}`

```
\begin{question}[ID=first]\label{qu:question_with_solution}
 This is our first question that gets a solution!
\end{question}
\begin{solution}
 This is the solution to exercise~\ref{qu:question_with_solution}!
\end{solution}
```

Exercise 8.
This is our first question that gets a solution!

You can see that in the default settings the solution is *not* written to the document. It has been saved, though, for possible later usage. We will see the solution later!

8.4 Options to the `solution` Environment

The `solutions` environment also has options, namely these:

`solution` ► `name = <name>` (initially empty)
sets a custom name.

`solution` ► `print = true|false` Default: `false`
prints or hides the solution.

Their meaning is the same as those for the `question` environment.

```
\begin{question}{5}
 The solution to this questions gets printed where it is.
\end{question}
\begin{solution}[print]
 See? This solution gets printed where you have put it in the code of
 your document.
\end{solution}
\begin{question}{2.5}
 The solution to this questions gets printed where it is \emph{and}
 has a fancy name. Have you noticed that you can assign partial
 points?
\end{question}
\begin{solution}[print,name=Fancy name]
 See? This solution gets printed where you have put it and has a fancy
 name!
\end{solution}
```

Exercise 9. 5 P.

The solution to this questions gets printed where it is.

Solution 9.

See? This solution gets printed where you have put it in the code of your document.

Exercise 10. 2.5 P.

The solution to this questions gets printed where it is *and* has a fancy name. Have you noticed that you can assign partial points?

Fancy name 10.

See? This solution gets printed where you have put it and has a fancy name!

8.5 Setting the Counter

The package option `counter-format` allows you to specify how the question counter is formatted.

The input is an arbitrary string which means you can have anything as counter number. However, the letter combinations `ch`, `se`, `qu` and `tsk` are replaced with the counters for the chapter, section, question or tasks (see part III), respectively. While the last one is not really useful in this case the others allow for a combined numbering. Each of these letter combinations can have an optional argument that specifies the format of the respective counter. `1`: `\arabic`, `a`: `\alph`, `A`: `\Alph`, `r`: `\roman` and `R`: `\Roman`.

```
\SetupExSheets{counter-format=Nr~se~(qu[a])}
\begin{question}
 A question with a differently formatted number.
\end{question}
```

Exercise Nr 8 (k)
A question with a differently formatted number.

Since the strings associated with the counters are replaced one has to hide them if they are actually wanted in the counter format. The easiest way would to hide them in braces.

```
\SetupExSheets{counter-format={section}\,se~{question}\,(qu[a])}
\begin{question}
 A question with a yet differently formatted number.
\end{question}
```

Exercise section 8 question (l)
A question with a yet differently formatted number.

8.6 Language Settings

The names of the questions and solutions are language dependent. If you use babel[19] or polyglossia[20] ExSheets will adapt to the document language. Exsheets has a number of translations but surely not all! If you miss a language please drop me a line in an email[21] containing the babel language name and the correct translations for questions (possibly distinguishing between exercises and exam questions) and solutions.

Until I implement it you can add something like this to your preamble (example for Danish) and try if it works:

```
\DeclareTranslation{Danish}{exsheets-exercise-name}
  {\O{}velse}
\DeclareTranslation{Danish}{exsheets-question-name}
  {Opgave}
\DeclareTranslation{Danish}{exsheets-solution-name}
  {Opl\o{}sning}
```

If this isn't working it means that the language you're using is unknown to the translations package (see part IV). In this case please notify me, too. You then can still use the `name` options.

9 Counting Points

9.1 The Commands

You have seen in section 8.1 that you can assign points to a question. If you do so these points are printed into the margin[22] and are counted internally. But there are additional commands to assign points or bonus points and a number of commands to retrieve the sum of points and/or bonus points.

[19] CTAN: babel [20] CTAN: polyglossia [21] contact@mychemistry.eu [22] Well, not necessarily. It depends on the heading style you have chosen.

► `\addpoints*{<num>}`
This command can be used to add points assigned to subquestions. `\addpoints` will print the points (with "unit") *and* add them to the sum of all points, `\addpoints*` will only add them but print nothing.

► `\points*{<num>}`
This command will only print the points (with "unit") but won't add them to the sum of points.

► `\addbonus*{<num>}`
This command can be used to add bonus points assigned to subquestions. `\addbonus` will print the points (with "unit") *and* add them to the sum of all bonus points, `\addbonus*` will only add them but print nothing.

► `\bonus*{<num>}`
This command will only print the bonus points (with "unit") but won't add them to the sum of bonus points.

► `\pointssum*`
Prints the sum of all points with or without (starred version) "unit": 67.75 P.

► `\currentpointssum*`
Prints the current sum of points with or without (starred version) "unit": 11.5 P.

► `\bonussum*`
Prints the sum of all bonus points with or without (starred version) "unit": 4 P.

► `\currentbonussum*`
Prints the current sum of bonus points with or without (starred version) "unit": 4 P.

► `\totalpoints*`
prints the sum of the points *and* the sum of the bonus points with "unit": 67.75 (+4) P. The starred version prints the sum of the points without "unit": 67.75 (+4).

The commands `\pointssum`, `\bonussum` and `\totalpoints` need at least *two* LaTeX runs to get the sum right.

Suppose you have an exercise worth 4 P. which consists of four questions listed with an `enumerate` environment that are all worth 1 P. each. You have two possibilities to display and count them:

```
% uses package 'enumitem'
\begin{question}{4}
 \begin{enumerate}[label=\alph*)]
  \item blah (\points{1})
  \item blah (\points{1})
  \item blah (\points{1})
  \item blah (\points{1})
 \end{enumerate}
\end{question}
\begin{question}
 \begin{enumerate}[label=\alph*)]
  \item blah (\addpoints{1})
  \item blah (\addpoints{1})
  \item blah (\addpoints{1})
  \item blah (\addpoints{1})
 \end{enumerate}
\end{question}
```

Exercise 13. 4 P.

a) blah (1 P.)

b) blah (1 P.)

c) blah (1 P.)

d) blah (1 P.)

Exercise 14.

a) blah (1 P.)

b) blah (1 P.)

c) blah (1 P.)

d) blah (1 P.)

9.2 Options

`points` ▶ `name = <name>` Default: `P.`

Choose the "unit" for the points. If you like to differentiate between a single point and more than one point you can give a plural ending separated with a slash: `name = point/s`. This sets also the name of the bonus points.

`points` ▶ `name-plural = <plural form of name>` (initially empty)

Instead of forming the plural form with an ending to the singular form this option allows to set an extra word for it. This sets also the plural form for the bonus points.

`points` ▶ `bonus-name = <name>` Default: `P.`

Choose the "unit" for the bonus points. If you like to differentiate between a single point and more than one point you can give a plural ending separated with a slash: `bonus-name = point/s`.

`points` ▶ `bonus-plural = <plural form of name>` (initially empty)

Instead of forming the plural form with an ending to the singular form this option allows to set an extra word for it.

`points` ▶ `use-name = true|false` Default: `true`

Don't display the name at all. Or do.

`points` ▶ `number-format = <any code>` (initially empty)

This option allows formatting of the number, e.g. italics: `number-format = \textit`.

`points` ▶ `bonus-format = <any code>` (initially empty)

This option allows formatting of the number of the bonus points, e.g. italics: `bonus-format = \textit`.

points ▶ parse = true|false Default: true
If set to false the points are not counted and the \totalpoints, \pointssum and \bonussum commands won't know their value.

points ▶ separate-bonus = true|false Default: false
This option determines whether points and bonus points each get their own unit when they appear together (in the margin or with \totalpoints).

points ▶ pre-bonus = <tokens> Default: (+
Code to be inserted before the bonus points when they follow normal points.

points ▶ post-bonus = <tokens> Default:)
Code to be inserted after the bonus points when they follow normal points.

```
\SetupExSheets[points]{name=point/s,number-format=\color{red}}
\begin{question}{1}
 This one's easy so only 1 point can be earned.
\end{question}
\begin{question}{7.5}
 But this one's hard! 7.5 points are in there for you!
\end{question}
```

Exercise 15. 1 point
This one's easy so only 1 point can be earned.

Exercise 16. 7.5 points
But this one's hard! 7.5 points are in there for you!

10 Printing Solutions

You have already seen that you can print solutions where they are using the print option. But ExSheets offers you quite more possibilities.

In the next subsections the usage of the command

▶ \printsolutions[<setting>]

is discussed.

Before we do that a hint: remember that you can set the option print globally:

```
% in the preamble
\SetupExSheets{solution/print=true}
```

Now if you want to typeset some text depending on the option being true or not you can use the following commands:

▷ `\PrintSolutionsTF{<true code>}{<false code>}`

▷ `\PrintSolutionsT{<true code>}`

▷ `\PrintSolutionsF{<false code>}`

They might come in handy if you want two versions of an exercise sheet, one with the exercises and one with the solutions, and you want to add different titles to these versions, for instance.

10.1 Print all

The first and easiest usage of `\printsolutions` is the following:

```
\printsolutions
```

There is nothing more to say, really. It prints all solutions you have specified except those belonging to a question with option `use = false`. Yes, there's one more point: `\printsolutions` only knows the solutions that have been set *before* its usage! This is also true for every usage explained in the next sections.

```
\printsolutions
```

Solution 8.
This is the solution to exercise 8!

Solution 9.
See? This solution gets printed where you have put it in the code of your document.

Fancy name 10.
See? This solution gets printed where you have put it and has a fancy name!

10.2 Print per chapter/section

Current chapter/section

If you are not creating an exercise sheet or an exam but are writing a textbook you maybe want a section at the end of each chapter showing the solution to the exercises presented in that chapter. In this case use the command as follows:

```
\printsolutions[section]
% or
\printsolutions[chapter]
```

Again, this is pretty much self-explaining. The solutions to the questions of the current chapter[23] or section are printed.

```
\begin{question}
 This is the first and only question in this section.
\end{question}
\begin{solution}
 This will be one of a few solutions printed by the following call of
 \cmd{printsolutions}.
\end{solution}
And now:
\printsolutions[section]
```

Exercise 17.
This is the first and only question in this section.

And now:

Solution 17.
This will be one of a few solutions printed by the following call of `\printsolutions`.

Specific chapter/section

You can also print only the solutions from chapters or sections other than the current ones. The syntax is fairly easy:

```
\printsolutions[section={1-7,10}]
% the same for chapters:
% \printsolutions[chapter={1-7,10}]
```

Solution 17.
This will be one of a few solutions printed by the following call of `\printsolutions`.

Don't forget that `\printsolutions` cannot know the solutions from section 10 yet. It is just used to demonstrate the syntax. You can also use an open range, e.g. something like

```
\printsolutions[section={-4,10-}]
```

This would print the solutions from sections 1–4 and from all sections with number 10[24] and greater.

There is an obvious disadvantage: you have to know the section numbers! But there is a solution: use the package option `use-ref = true`. Then you can do something like

[23] Only if the document class you're using *has* chapters, of course!

[24] Or rather where `\value{section}` is 10 or greater – the actual counter formatting is irrelevant.

```
% in the preamble:
\usepackage[use-ref]{exsheets}
% somewhere in your code after \section{A really cool section
  title}:
\label{sec:ReallyCool}
% somewhere later in your code:
\printsolutions[section={-\S{sec:ReallyCool}}]
% which will print all solutions from questions up to and
% including the really cool section
```

With the package option `use-ref = true` each usage of `\label` will create additional labels (one preceded with `exse:` and another one with `exch:`) which store the section number and the chapter number, respectively. These are used internally by two commands `\S` and `\C` which refer to the section number and the chapter number the label was created in. *These commands are only available as arguments of* `\printsolutions`.

Since some packages like the well known hyperref[25] for example redefine `\label` `use-ref` won't work in together with it. In this case don't use `use-ref` and set `\exlabel{<label>}` instead to remember the section/the chapter number. Its usage is just like `\label`. So the safest way is as follows:

```
% in the preamble:
\usepackage{exsheets}
% somewhere in your code after \section{A really cool section
  title}:
\exlabel{sec:ReallyCool}
% somewhere later in your code:
\printsolutions[section={-\S{sec:ReallyCool}}]
% which will print all solutions from questions up to and
% including the really cool section
```

Please be aware that the labels must be processed in a previous LaTeX run before `\S` and `\C` can pass them on to `\printsolutions`.

10.3 Print by ID

Now comes the best part: you can also print selected solutions! Every question has an ID. To see which ID a question has you can call the following command:

▶ `\DebugExSheets{true|false}`

▶ `\CurrentQuestionID`

Introduced in version 0.4a

expands to the current question ID (after two expansions).

Let's create some more questions and take a look what this command does:

[25] CTAN: hyperref

```
\DebugExSheets{true}
\begin{question}[ID=nice!]
 A question with a nice \acs{id}!
\end{question}
\begin{solution}
 The solution to the question with the nice \acs{id}.
\end{solution}
\begin{question}{3.75}
 Yet another question. But this time with quarter points!
\end{question}
\begin{solution}
 Yet another solution.
\end{solution}
```

ID: *nice!*

Exercise 18.

A question with a nice ID!

ID: *19*

Exercise 19. 3.75 P.

Yet another question. But this time with quarter points!

So now we can call some specific solutions:

```
\printsolutions[byID={first,nice!,10,14}]
```

Solution 8.

This is the solution to exercise 8!

Fancy name 10.

See? This solution gets printed where you have put it and has a fancy name!

Solution 18.

The solution to the question with the nice ID.

This makes use of the l3sort package which at the time of writing is still considered experimental. In case you wonder where solution 14 is: question 14 has no solution given.

If you don't want that the solutions are sorted automatically but appear in the order given you can use the option

`solution` ▶ `sorted` = `true`|`false` Default: `true`

Sort solutions given by ID or don't.

11 Dividing Questions into Classes

11.1 Using Classes

For creating different variants of a written exam or different difficulty levels of an exercise sheet it comes in handy if one can assign certain classes to questions and then tell ExSheets only to use one ore more specific classes.

▶ `use-classes = <list of classes>` (initially empty)
When this option is used only the questions belonging to the specified classes are printed and have their solutions saved.

```
\SetupExSheets{use-classes={A,C}}
\begin{question}[class=A]
 Belonging to class A.
\end{question}
\begin{question}[class=B]
 Belonging to class B.
\end{question}
\begin{question}[class=C]
 Belonging to class C!
\end{question}
```

Exercise 20.
Belonging to class A.

Exercise 21.
Belonging to class C!

Questions of classes that are not used are fully discarded. *This also means that questions that don't have a class assigned are discarded.*

11.2 Using Topics

Similarly to classes one can assign topics to questions. The usage is practically identical, the semantic meaning is different.

▶ `use-topics = <list of topics>` (initially empty)
When this option is used only the questions belonging to the specified topics are printed and have their solutions saved.

```
\SetupExSheets{use-topics={trigonometry}}
\begin{question}[topic=trigonometry]
 A trigonometry question.
\end{question}
\begin{question}[topic=arithmetics]
 A arithmetics question
\end{question}
```

Exercise 22.
A trigonometry question.

Questions of topics that are not used are fully discarded. *This also means that questions that don't have a topic assigned are discarded.*

If you set both `use-classes` and `use-topics` then only questions will be used that *match both categories.*

Ideally one could assign more than one topic to a question but this is *not* supported yet.

11.3 Own Dividing Concepts

Introduced in version 0.8

Actually both classes and topics are introduced into ExSheets internally this way:

```
\DeclareQuestionClass{class}{classes}
\DeclareQuestionClass{topic}{topics}
```

which means you can do the same introducing your own dividing concepts.

▶ `\DeclareQuestionClass{<singular name>}{<plural name>}`

Introduces a new dividing concept and defines both new options for the `question` environment and new global options.

For example you could decide you want to group your questions according to their difficulty. You could place the following line in your preamble:

```
\DeclareQuestionClass{difficulty}{difficulties}
```

This would define an option `use-difficulties` analogous to `use-classes` and `use-topics`. It would also define an option `difficulty` for the `question` environment. This means you could now do something like the following:

```
\SetupExSheets{use-difficulties={easy,hard}}
\begin{question}[difficulty=easy]
 An easy question.
\end{question}
\begin{question}[difficulty=medium]
 This one's a bit harder.
\end{question}
\begin{question}[difficulty=hard]
 Now let's see if you can solve this one.
\end{question}
```

Exercise 23.
An easy question.

Exercise 24.
Now let's see if you can solve this one.

12 Adding and Using Additional Information to Questions

For managing lots of questions and corresponding solutions it can be very useful to be able to save and recover additional information to the questions. This is possible with the following commands. First the ones for saving:

- `\DeclareQuestionProperty{<name>}`
  This command defines a question property `<name>`. It can only be used in the document preamble.

- `\SetQuestionProperties{<name>=<value>,...}`
  Set the properties for a specific question. this command can only be used inside the `question` environment.

Now the commands for recovering the properties:

- `\QuestionNumber{<id>}`
  Recover the number of the question with the ID `<id>`. The number is displayed according to the format set with `counter-format`.

- `\GetQuestionProperty{<name>}{<id>}`
  Recover the property `<name>` of the question with the ID `<id>`. Of course the property must have been declared before.

Let's say we have declared the properties `notes`, `reference` and `topic`. By default the property `points` is available and gets the value of the optional argument of the `question` environment.

We can now do the following:

```
% uses 'biblatex'
\begin{question}[ID=center,topic=LaTeX]{3}
 Explain how you could center text in a \LaTeX\ document.
 \SetQuestionProperties{
   topic     = \TeX/\LaTeX ,
   notes     = {How to center text.},
   reference = {\textcite{companion}}}
\end{question}
\begin{solution}
 To center a short part of the text body one can use the \texttt{center}
 environment (\points{1}). Inside an environment like \texttt{table} one
 should use \texttt{\string\centering} (\points{1}). For single lines
 there is also the \texttt{\string\centerline} command (\points{1}).
\end{solution}
\begin{question}[ID=knuthbooks,topic=LaTeX]{2}
 Name two books by D.\,E.\,Knuth.
 \SetQuestionProperties{
   topic     = \TeX/\LaTeX ,
   notes     = {Books by Knuth.},
   reference = {\textcite{knuth:ct:a,knuth:ct:b,knuth:ct:c,knuth:ct:d,
  knuth:ct:e}}}
\end{question}
\begin{solution}
 For example two volumes from \citetitle{knuth:ct}:
 \citetitle{knuth:ct:a,knuth:ct:b,knuth:ct:c,knuth:ct:d,knuth:ct:e}. Each
  valid
 answer is worth \points{1}
\end{solution}
```

Exercise 25. 3 P.

Explain how you could center text in a LaTeX document.

Exercise 26. 2 P.

Name two books by D. E. Knuth.

It is now possible to recover these values later:

```
% uses 'booktabs'
\begin{center}
 \begin{tabular}{lll}
  \toprule
   Question & Property & \\
  \midrule
   \QuestionNumber{center}
     & Points     & \GetQuestionProperty{points}{center} \\
     & Topic      & \GetQuestionProperty{topic}{center} \\
     & References & \GetQuestionProperty{reference}{center} \\
     & Note       & \GetQuestionProperty{notes}{center} \\
  \midrule
   \QuestionNumber{knuthbooks}
     & Points     & \GetQuestionProperty{points}{knuthbooks} \\
     & Topic      & \GetQuestionProperty{topic}{knuthbooks} \\
     & References & \GetQuestionProperty{reference}{knuthbooks} \\
     & Note       & \GetQuestionProperty{notes}{knuthbooks} \\
  \bottomrule
 \end{tabular}
\end{center}
```

Question	Property	
25.	Points	3
	Topic	TEX/LATEX
	References	Goossens, Mittelbach, and Samarin [3]
	Note	How to center text.
26.	Points	2
	Topic	TEX/LATEX
	References	Knuth [7, 8, 9, 10, 11]
	Note	Books by Knuth.

Introduced in version 0.7a

If you use the package option `auto-label` the properties `ref` and `pageref` are predefined which will call the corresponding `\ref` or `\pageref`, respectively.

Please note that these properties *are not the same* as the dividing concepts explained in section 11 although they may seem similar in meaning or even have the same name.

Introduced in version 0.3

There are additional commands that might prove useful:

▶ `\ForEachQuestion{<code to be executed for each used question>}`

Inside the argument one can refer to the ID of a question with `#1`. Beware that this command only knows of questions used before it is issued.

▷ `\numberofquestions`

Returns the current number of used questions. Beware that this command only knows of questions used before it is issued.

▷ `\iflastquestion{<true code>}{<false code>}`

Although this command is available in the whole document it is only useful inside \ForEachQuestion. It tells you if the end of the loop is reached or not.

For example one could use these commands to create a grading table:

```
\begin{tabular}{|l|*{\numberofquestions}{c|}c|}\hline
 Question &
   \ForEachQuestion{\QuestionNumber{#1}\iflastquestion{}{&}} &
   Total \\ \hline
 Points   &
   \ForEachQuestion{\GetQuestionProperty{points}{#1}\
  iflastquestion{}{&}} &
   \pointssum* \\ \hline
 Reached  &
   \ForEachQuestion{\iflastquestion{}{&}} & \\ \hline
\end{tabular}
```

For four questions the table now would look similar to figure 1.

Question	1.	2.	3.	4.	Total
Points	3	5	10	8	26
Reached					

Figure 1: An example for a grading table. (Actually this is a fake. See the `grading-table.tex` file shipped with exsheets for the real use case.)

13 Variations of an Exam

Introduced in version 0.6

It is a quite common task to design an exam in two different variants. This is of course possible with ExSheets' classes (see section 11.1). However, often not the whole question is to be different but only small details, the numbers in a maths exam, say. For this purpose ExSheets provides the following commands:

- \SetVariations{<num>}
  Set the number of different variants. This will determine how many arguments the command \vary will get. <num> must at least be 2 and is initially set to 2.

- \variant{<num>}
  Choose the active variant. The argument must be a number between 1 and the number set with \SetVariations. Initially set to 1.

- \vary{<variant 1>}{<variant 2>}
  This command is the one actually used in the document. It has a number of required arguments equal to the number set with \SetVariations. All of its arguments are discarded except the one specified with \variant.

▶ \lastvariant

Introduced in version 0.7b

Each time \vary is called it stores the value it chose in \lastversion. This might be convenient to use if one otherwise would have to repeatedly write the same \vary.

```
\SetVariations{6}%
\variant{6}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
\variant{1}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
\variant{5}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
\variant{2}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
\variant{4}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
\variant{3}\vary{A}{B}{C}{D}{E}{F}
(last variant: \lastvariant)
```

F (last variant: F) A (last variant: A) E (last variant: E) B (last variant: B) D (last variant: D) C (last variant: C)

14 A Grade Distribution

Probably this is a rather esoteric feature but it could proof useful in some cases. Suppose you are a German math teacher and want to grade exactly corresponding to the number of points relative to the sum of total points, regardless of how big that might be. You could do something like this to present your grading decisions for the exam:

```
% preamble:
% \DeclareRelGrades{
%   1     = 1 ,
%   {1,5} = .9167 ,
%   2     = .8333 ,
%   {2,5} = .75 ,
%   3     = .6667 ,
%   {3,5} = .5833 ,
%   4     = .5
% }
\small\setlength\tabcolsep{2pt}
\begin{tabular}{r|*8c}
 Punkte
   & $\grade*{1}$      & $\le\grade*{1}$ & $\le\grade*{1,5}$ & $\le\grade
 *{2}$
   & $\le\grade*{2,5}$ & $\le\grade*{3}$ & $\le\grade*{3,5}$ & $<\grade
 *{4}$ \\
 Note
   & 1 & 1--2 & 2 & 2--3 & 3 & 3--4 & 4 & 5
\end{tabular}
```

Punkte	$67.75 \leq$	$67.75 \leq$	$62 \leq$	$56 \leq$	$51 \leq$	$45 \leq$	$40 <$	34
Note	1	1–2	2	2–3	3	3–4	4	5

These are the available commands and options:

▶ `\DeclareRelGrades{<grade>=<num>,...}`
This command is used to define grades and assign the percentage of total points to them.

▶ `\grade*{<grade>}`
Gives the number of points corresponding to a grade depending on the value of `\pointssum` with or without (starred version) "unit".

`grades` ▶ `round = <num>` Default: `0`
The number of decimals the points of a grade are rounded to. This doesn't apply to the maximum number of points if the rounded number would be bigger than the actual sum.

`grades` ▶ `half = true|false` Default: `false`
If set to `true` points are rounded either to full or to half points.

15 Selectively Include Questions from External Files

15.1 Caveat

I need to say some words of caution: the `\includequestions` that will be presented shortly is probably ExSheets' most experimental one at the time of writing (April 21, 2013). Thanks to feedback of users it is constantly improved and bugs are fixed. It is not a very efficient way to insert question regarding performance and you shouldn't wonder if compilation slows down when you use it. It probably needs to be re-written all over but on the one hand that would introduce new bugs and on the other hand for the time being I don't have the capacities, anyway, so you'll have to live it, I'm afraid.

15.2 How it works

Suppose you have one or more files with questions prepared to use them as a kind of database. One for class A, say, one for class B, one for class C and so one, something like this:

```
% this is file classA.tex
\begin{question}[class=A,ID=A1,topic=X]
 First question of class A, topic X.
\end{question}
\begin{solution}
 First solution of class A.
\end{solution}
\begin{question}[class=A,ID=A2,topic=Y]
 Second question of class A, topic Y.
\end{question}
\begin{solution}
 Second solution of class A.
```

```
\end{solution}
...
% end of file classA.tex
\endinput
```

You can of course just \input or \include it but that would of course include the whole file into your document. But would't it be nice to just include selected questions? Or maybe a five random questions from the file? That is possible with the following command:

▶ \includequestions[<options>]{<list of filenames>}

If you use it without options it will have the same effect as \input. There are however the following options:

include ▶ all = true|false

include ▶ IDs = <list of IDs> (initially empty)

Includes only the specified questions.

include ▶ random = <num> (initially empty)

Includes <num> randomly selected questions. This option uses the pgfcore package to create the pseudo-random numbers.

include ▶ exclude = <list of IDs> (initially empty)

Questions who's IDs are specified here are *not* included. This option can be combined with the random option.

The usage should be self-explainable:

```
% include questions A1, A3 and A4:
\includequestions[IDs={A1,A3,A4}]{classA.tex}
% or include 3 random questions:
\includequestions[random=3]{classA}
```

In order to be able to select the questions ExSheets needs to \input the file twice. The first time the available questions are determined, the second time the selected questions are used. This unfortunately means that anything that is *not* part of a question or solution is also input twice. Either don't put anything else into the file or use one of the following commands for control:

▶ \PrintIfIncludeActiveTF{<true code>}{<false code>}

▶ \PrintIfIncludeActiveT{<true code>}

▶ \PrintIfIncludeActiveF{<false code>}

The selection can be refined further by selecting questions belonging to a specific class of questions (see section 11) before using `\includequestions`.

Introduced in version 0.8

After you've used `\includequestions` the IDs of the included questions is available as an unordered comma separated list in the following macro:

- `\questionsincludedlast`
  Unordered comma separated list of question IDs included with the last usage of `\includequestions`.

16 Own Question/Solution Pairs

Changed in version 0.9

ExSheets provides he possibility to create new environments that behave like the `question` and `solution` environments. This would allow, for example, to define a `question*`/`solution*` environment pair for bonus questions. The following commands may be used in the document preamble:

- `\NewQuSolPair{<question>}[<question options>][<general options>]{<solution>}[<solution options>][<general options>]`
- `\RenewQuSolPair{<question>}[<question options>][<general options>]{<solution>}[<solution options>][<general options>]`

The standard environments are defined as follows:

```
\NewQuSolPair{question}{solution}
```

Let's say we want the possibility to add bonus questions. A simple way would be to define starred variants that add a star in the margin left to the title:

```
% preamble:
% \NewQuSolPair
%   {question*}[name=\protect\llap{$\bigstar$\space}Bonus Question]
%   {solution*}[name=\protect\llap{$\bigstar$\space}Solution]
\begin{question*}
 This is a bonus question.
\end{question*}
\begin{solution*}[print]
 This is what the solution looks like.
\end{solution*}
```

★ Bonus Question 27.
This is a bonus question.

★ Solution 27.
This is what the solution looks like.

As you can see the environments take the same options as are described for the standard `question` and `solution` environments.

17 Filling in the Blanks

17.1 Cloze

Changed in version 0.4

Both in exercise sheets and in exams it is sometimes desirable to be able to create _____ that have to be filled in. Or may be some more lines: _____

_____

_____

_____

_____

_____

▶ `\blank*[<options>]{<text to be filled in>}`
creates a blank in normal text or in a question but fills the text of its argument if inside a solution. If used at the *begin of a paragraph* `\blank` will do two things: it will set the linespread according to an option explained below and will insert `\par` after the lines. If you don't want that use the starred version.

The options are these:

`blank` ▶ `style = line|wave|dline|dotted|dashed` Default: `line`
The style of the line. This uses the corresponding command from the ulem package and is the whole reason why ExSheets loads it in the first place.

`blank` ▶ `scale = <num>` Default: 1
Scales the width of the blank by factor `<num>` unless the width is explicitly set.

`blank` ▶ `width = <dim>` (initially empty)
The width of the line. If it is not used the width of the filled in text is used.

`blank` ▶ `linespread = <num>` Default: 1
Set the linespread for the blank lines. This only has an effect if `\blank` is used at the begin of a paragraph.

```
\begin{question}
 Try to fill in \blank[width=4cm]{these} blanks. All of them
 \blank[style=dotted]{are created} by using the \cmd{blank}
 \blank[style=dashed]{command}.
\end{question}
\begin{solution}[print]
 Try to fill in \blank[width=4cm]{these} blanks. All of them
 \blank[style=dotted]{are created} by using the \cmd{blank}
 \blank[style=dashed]{command}.
\end{solution}
```

Exercise 28.
Try to fill in _____ blanks. All of them ..... by using the \blank _____.

Solution 28.
Try to fill in these blanks. All of them are created by using the \blank command.

A number of empty lines are easily created by setting the width option:

```
\blank[width=4.8\linewidth,linespread=1.5]{}
```

17.2 Vertical Space for answers

Introduced in version 0.3

When you're creating an exam you might want to add some vertical space where the students can write down their answers. While you can always use \vspace this is not always handy when the space left on the page is less than you want. In this case it would be nice if a) there would be no warning and b) the rest of the space would be added at the top of the next page. This is what the following command is for:

► \examspace*{<dim>}
Add space as specified in {<dim>}. If the space available on the current page is not enough the rest of the space will be added at the top of the next page. The starred version will silently drop any leftover space instead of adding it to the next page.

```
\begin{question}
 What do you think of this feature?
 \examspace{3cm}
\end{question}
This line comes after the space.
```

Exercise 29.
What do you think of this feature?

This line comes after the space.

18 Styling your Exercise/Exam Sheets

18.1 Background

The ExSheets package makes extensive use of LaTeX3's coffins[26] as well as its templates concept.[27] The latter allows a rather easy extension and customization of some of ExSheets' environments. To be more precise: you can define your own instances for the headings used for questions and solutions and for the `tasks` environment.

What this package doesn't provide is changing the background of questions or framing them. But this is easily possible using the mdframed[28] package and its `\surroundwithmdframed` command.

18.2 The `exsheets-headings` Object

ExSheets defines the object `exsheets-headings` and one template for it, the 'default' template. The package also defines two instances of this template, the 'block' instance and the 'runin' instance.

```
\SetupExSheets{headings=block}
\begin{question}{1}
 a 'block' heading
\end{question}
\SetupExSheets{headings=runin}
\begin{question}{1}
 a 'runin' heading
\end{question}
```

Exercise 30. 1 P.
a 'block' heading

Exercise 31. a 'runin' heading 1 P.

[26] See the documentation to the xcoffins package for more information on that. [27] Have a look into the documentation to the xtemplate package. [28] CTAN: mdframed

18.2.1 Available Options

This section only lists the options that can be used when defining an instance of the 'default' template. The following subsections will give loads of examples of their usage. The options are listed in the definition for the template interface:

```
\DeclareTemplateInterface{exsheets-heading}{default}{3}
 {
   % option         : type      = default
   inline           : boolean   = false ,
   runin            : boolean   = false ,
   indent-first     : boolean   = false ,
   toc-reversed     : boolean   = false ,
   vscale           : real      = 1     ,
   above            : length    = 2pt   ,
   below            : length    = 2pt   ,
   main             : tokenlist =       ,
   pre-code         : tokenlist =       ,
   post-code        : tokenlist =       ,
   title-format     : tokenlist =       ,
   title-pre-code   : tokenlist =       ,
   title-post-code  : tokenlist =       ,
   number-format    : tokenlist =       ,
   number-pre-code  : tokenlist =       ,
   number-post-code : tokenlist =       ,
   points-format    : tokenlist =       ,
   points-pre-code  : tokenlist =       ,
   points-post-code : tokenlist =       ,
   join             : tokenlist =       ,
   attach           : tokenlist =
 }
```

Each heading is built with at most four coffins available with the names 'main', 'title', 'number' and 'points'. Those coffins place possibly the whole heading, the title, the question number and the assigned points. The only coffin that's always typeset is the 'main' coffin, which is empty per default.

Coffins can be joined (two become one, the first extends its bounding box to contain the second) using the following syntax:

```
join = coffin1[handle11,handle12]coffin2[handle21,handle22](x-
  offset,y-offset)
```

The syntax for attaching (two become one, the first does *not* extend its bounding box around the second) is the same.

More on coffin handles is described in the documentation for the xcoffins. Figure 2 briefly demonstrates the available handle pairs.

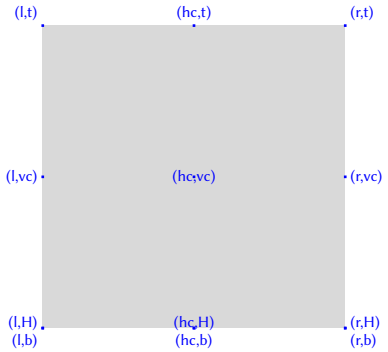

FIGURE 2: Available handles for a horizontal coffin.

The following subsections will show all definitions of the instances available with the package option `load-headings` and how they look. This will hopefully give you enough ideas to create your own instance if you want to have another heading style than the ones available.

Of you use the option `load-headings` each of the following instances is available through the option `headings = <instance>`.

The following examples use a sample text defined as follows:

```
\def\s{This is some sample text we will use to create a
  somewhat
  longer text spanning a few lines.}
\def\sample{\s\ \s\par\s}
```

18.2.2 The 'block' Instance

```
\DeclareInstance{exsheets-heading}{block}{default}
 {
   join             = { title[r,B]number[l,B](1ex,0pt) } ,
   attach           =
     {
       main[l,vc]title[l,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](\marginparsep,0pt)
     }
 }
```

Exercise 32. 1 P.

A 'block' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.3 The 'runin' Instance

```
\DeclareInstance{exsheets-heading}{runin}{default}
 {
   runin            = true ,
   number-post-code = \space ,
   attach           =
     { main[l,vc]points[l,vc](\linewidth+\marginparsep,0pt) }
 ,
   join             =
     {
       main[r,vc]title[r,vc](0pt,0pt) ;
       main[r,vc]number[l,vc](1ex,0pt)
     }
 }
```

Exercise 33. A 'runin' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.4 The 'simple' Instance

```
\DeclareInstance{exsheets-heading}{simple}{default}
 {
   title-format     = \normalsize ,
   points-pre-code  = ( ,
   points-post-code = ) ,
   attach           = { main[l,t]number[l,t](0pt,0pt) } ,
   join             =
     {
       number[r,b]title[l,b](1ex,0pt) ;
       main[l,b]points[l,t](1em,0pt)
     }
 }
```

34. Exercise
(1 P.)

A 'simple' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.5 The ‘empty’ Instance

Introduced in version 0.9a

```
\DeclareInstance{exsheets-heading}{empty}{default}
 {
   runin  = true ,
   above  = \parskip ,
   below  = \parskip ,
   attach = { main[l,vc]points[l,vc](\linewidth+\marginparsep
  ,0pt) }
 }
```

An ‘empty’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.6 The ‘block-rev’ Instance

```
\DeclareInstance{exsheets-heading}{block-rev}{default}
 {
   toc-reversed     = true ,
   join             = { number[r,B]title[l,B](1ex,0pt) } ,
   attach           =
     {
       main[l,vc]number[l,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](\marginparsep,0pt)
     }
 }
```

36. Exercise 1 P.

A ‘block-rev’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.7 The ‘block-wp’ Instance

```
\DeclareInstance{exsheets-heading}{block-wp}{default}
 {
   points-pre-code  = ( ,
   points-post-code = ) ,
   join             =
     {
       title[r,B]number[l,B](1ex,0pt) ;
```

```
      title[r,B]points[l,B](1ex,0pt)
    } ,
  attach           = { main[l,vc]title[l,vc](0pt,0pt) }
}
```

Exercise 37. (1 P.)
A 'block-wp' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.8 The 'block-wp-rev' Instance

```
\DeclareInstance{exsheets-heading}{block-wp-rev}{default}
 {
   toc-reversed     = true ,
   points-pre-code  = ( ,
   points-post-code = ) ,
   join             =
     {
       number[r,B]title[l,B](1ex,0pt) ;
       number[r,B]points[l,B](1ex,0pt)
     } ,
   attach           = { main[l,vc]number[l,vc](0pt,0pt) }
 }
```

38. Exercise (1 P.)
A 'block-wp-rev' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.9 The 'block-nr' Instance

```
\DeclareInstance{exsheets-heading}{block-nr}{default}
 {
   attach           =
     {
       main[l,vc]number[l,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](\marginparsep,0pt)
     }
 }
```

39. 1 P.
A 'block-nr' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.10 The 'block-nr-wp' Instance

```
\DeclareInstance{exsheets-heading}{block-nr-wp}{default}
 {
   points-pre-code  = ( ,
   points-post-code = ) ,
   join             = { number[r,vc]points[l,vc](1ex,0pt) } ,
   attach           = { main[l,vc]number[l,vc](0pt,0pt) }
 }
```

40. (1 P.)
A 'block-nr-wp' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.11 The 'runin-rev' Instance

```
\DeclareInstance{exsheets-heading}{runin-rev}{default}
 {
   toc-reversed     = true ,
   runin            = true ,
   title-post-code  = \space ,
   attach           =
     { main[l,vc]points[l,vc](\linewidth+\marginparsep,0pt) }
 ,
   join             =
     {
       main[r,vc]number[r,vc](0pt,0pt) ;
       main[r,vc]title[l,vc](1ex,0pt)
     }
 }
```

41. Exercise A 'runin-rev' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.12 The 'runin-wp' Instance

```
\DeclareInstance{exsheets-heading}{runin-wp}{default}
 {
   runin            = true ,
   points-pre-code  = ( ,
   points-post-code = )\space ,
   join             =
     {
       main[r,vc]title[r,vc](0pt,0pt) ;
       main[r,vc]number[l,vc](1ex,0pt) ;
       main[r,vc]points[l,vc](1ex,0pt)
     }
 }
```

Exercise 42. (1 P.) A 'runin-wp' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.13 The 'runin-wp-rev' Instance

```
\DeclareInstance{exsheets-heading}{runin-wp-rev}{default}
 {
   toc-reversed     = true ,
   runin            = true ,
   points-pre-code  = ( ,
   points-post-code = )\space ,
   join             =
     {
       main[r,vc]number[r,vc](0pt,0pt) ;
       main[r,vc]title[l,vc](1ex,0pt) ;
       main[r,vc]points[l,vc](1ex,0pt)
     }
 }
```

43. Exercise (1 P.) A 'runin-wp-rev' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.14 The 'runin-nr' Instance

```
\DeclareInstance{exsheets-heading}{runin-nr}{default}
 {
   runin            = true ,
   number-post-code = \space ,
   attach           =
     { main[l,vc]points[l,vc](\linewidth+\marginparsep,0pt) }
 ,
   join             = { main[r,vc]number[l,vc](0pt,0pt) }
 }
```

44. A 'runin-nr' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.15 The 'runin-fixed-nr' Instance

```
\DeclareInstance{exsheets-heading}{runin-fixed-nr}{default}
 {
   runin            = true ,
   number-pre-code  = \hbox to 2em \bgroup ,
   number-post-code = \hfil\egroup ,
   attach           =
     { main[l,vc]points[l,vc](\linewidth+\marginparsep,0pt) }
 ,
   join             = { main[r,vc]number[l,vc](0pt,0pt) }
 }
```

45. A 'runin-fixed-nr' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.16 The 'runin-nr-wp' Instance

```
\DeclareInstance{exsheets-heading}{runin-nr-wp}{default}
 {
   runin            = true ,
   points-pre-code  = ( ,
   points-post-code = )\space ,
   join             =
     {
       main[r,vc]number[l,vc](0pt,0pt) ;
```

```
      main[r,vc]points[l,vc](1ex,0pt)
    }
}
```

46. (1 P.) A ‘runin-nr-wp’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.17 The ‘inline’ Instance

Introduced in version 0.5

```
\DeclareInstance{exsheets-heading}{inline}{default}
 {
   inline           = true ,
   number-pre-code  = \space ,
   number-post-code = \space ,
   join             =
     {
       main[r,vc]title[r,vc](0pt,0pt) ;
       main[r,vc]number[l,vc](0pt,0pt)
     }
 }
```

Text before **Exercise 47.** An ‘inline’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines. Text after

18.2.18 The ‘inline-wp’ Instance

Introduced in version 0.5

```
\DeclareInstance{exsheets-heading}{inline-wp}{default}
 {
   inline           = true ,
   number-pre-code  = \space ,
   number-post-code = \space ,
   points-pre-code  = ( ,
   points-post-code = )\space ,
   join             =
     {
       main[r,vc]title[r,vc](0pt,0pt) ;
       main[r,vc]number[l,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](0pt,0pt)
```

```
      }
  }
```

Text before **Exercise 48.** (1 P.) An ‘inline-wp’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines. Text after

18.2.19 The ‘inline-nr’ Instance

Introduced in version 0.5

```
\DeclareInstance{exsheets-heading}{inline-nr}{default}
 {
   inline           = true ,
   number-post-code = \space ,
   join             = { main[r,vc]number[l,vc](0pt,0pt) }
 }
```

Text before **49.** An ‘inline-nr’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines. Text after

18.2.20 The ‘centered’ Instance

```
\DeclareInstance{exsheets-heading}{centered}{default}
 {
   join             = { title[r,B]number[l,B](1ex,0pt) } ,
   attach           =
     {
       main[hc,vc]title[hc,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](\marginparsep,0pt)
     }
 }
```

Exercise 50. 1 P.

A ‘centered’ heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.21 The 'centered-wp' Instance

```
\DeclareInstance{exsheets-heading}{centered-wp}{default}
 {
   points-pre-code  = ( ,
   points-post-code = ) ,
   join             =
     {
       title[r,B]number[l,B](1ex,0pt) ;
       title[r,B]points[l,B](1ex,0pt)
     } ,
   attach           = { main[hc,vc]title[hc,vc](0pt,0pt) }
 }
```

Exercise 51. (1 P.)

A 'centered-wp' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.22 The 'margin' Instance

```
\DeclareInstance{exsheets-heading}{margin}{default}
 {
   runin            = true ,
   number-post-code = \space ,
   points-pre-code  = ( ,
   points-post-code = )\space ,
   join             = { title[r,b]number[l,b](1ex,0pt) } ,
   attach           =
     {
       main[l,vc]title[r,vc](0pt,0pt) ;
       main[l,b]points[r,t](0pt,0pt)
     }
 }
```

Exercise 52. (1 P.)

A 'margin' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.23 The 'margin-nr' Instance

```
\DeclareInstance{exsheets-heading}{margin-nr}{default}
 {
   runin  = true ,
   attach =
     {
       main[l,vc]number[r,vc](-1ex,0pt) ;
       main[r,vc]points[l,vc](\linewidth+\marginparsep,0pt)
     }
 }
```

53. A 'margin-nr' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines. 1 P.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.24 The 'raggedleft' Instance

```
\DeclareInstance{exsheets-heading}{raggedleft}{default}
 {
   join             = { title[r,B]number[l,B](1ex,0pt) } ,
   attach           =
     {
       main[r,vc]title[r,vc](0pt,0pt) ;
       main[r,vc]points[l,vc](\marginparsep,0pt)
     }
 }
```

Exercise 54. 1 P.

A 'raggedleft' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.25 The 'fancy' Instance

```
\DeclareInstance{exsheets-heading}{fancy}{default}
 {
   toc-reversed     = true ,
   indent-first     = true ,
   vscale           = 2 ,
   pre-code         = \rule{\linewidth}{1pt} ,
   post-code        = \rule{\linewidth}{1pt} ,
```

```
    title-format     = \large\scshape\color{exsheetsred} ,
    number-format    = \large\bfseries\color{exsheetsblue} ,
    points-format    = \itshape ,
    join             = { number[r,B] title[l,B] (1ex,0pt) } ,
    attach           =
      {
        main[hc,vc]number[hc,vc](0pt,0pt) ;
        main[l,vc]points[r,vc](-\marginparsep,0pt)
      }
  }
```

1 P.

55. Exercise

A 'fancy' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.2.26 The 'fancy-wp' Instance

```
\DeclareInstance{exsheets-heading}{fancy-wp}{default}
 {
   toc-reversed     = true ,
   indent-first     = true ,
   vscale           = 2 ,
   pre-code         = \rule{\linewidth}{1pt} ,
   post-code        = \rule{\linewidth}{1pt} ,
   title-format     = \large\scshape\color{exsheetsred} ,
   number-format    = \large\bfseries\color{exsheetsblue} ,
   points-format    = \itshape ,
   points-pre-code  = ( ,
   points-post-code = ) ,
   join             =
     {
       number[r,B]title[l,B](1ex,0pt) ;
       number[r,B]points[l,B](1ex,0pt)
     } ,
   attach           = { main[hc,vc]number[hc,vc](0pt,0pt) }
 }
```

56. Exercise *(1 P.)*

A 'fancy-wp' heading. This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

18.3 Load Custom Configurations

If you have custom configurations you want to be loaded automatically then save them in a file `exsheets_configurations.cfg`. If this file is present it will be loaded `\AtBeginDocument`.

Part III
The TASKS package (v0.9)

19 Motivation

Changed in version 0.7

Originally TASKS has been an integral part of EXSHEETS. However, users told me that it indeed could be useful to have it as a standalone package not having to load the whole EXSHEETS beast just for having the `tasks` environment available. Since I agree with this the environment has been extracted into a package if its own, TASKS.

The reason for the `tasks` environment is an unwritten agreement in German maths textbooks (in (junior) high school, especially) to organize exercises in columns counting horizontally rather than vertically. That is what `tasks` primarily is for. If you don't need this feature you're better off using traditional LaTeX lists and the enumitem package for customization.

20 Requirements

TASKS requires packages l3kernel ,xparse, l3keys2e, epic,[29] cntformats,[30] xtemplate and environ.

21 How it works

21.1 The Basics

The `tasks` environment is similar to a list like `enumerate` but not the same:

- The first difference: there is no pagebreak possible inside an item but only between items.
- The second difference: the enumeration default is a), b), c) ...
- The third difference: there is a split at *every* occurrence of the item separator. For this reason the default separator is not `\item` but `\task` so it is unique to this environment only.
- The fourth difference: the `tasks` environment cannot be nested. You can, however, use an `itemize` environment or something in it.
- The fifth difference: verbatim material cannot be used in it. You'll have to use `\string`, `\texttt` or `\detokenize`. If this won't suffice then don't use `tasks`.

[29] CTAN: epic [30] Part of the EXSHEETS bundle, see part V

▶ `\begin{tasks}[<options>](<num of columns>)` `\task`... `\end{tasks}`

Let's see an example:

```
Some text before the list.
\begin{tasks}
 \task \sample
 \task \sample
 \task \sample
\end{tasks}
And also some text after it.
```

Some text before the list.

a) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

And also some text after it.

The environment takes the optional argument `(<num of columns>)` with which the number of columns used by the environment is specified.

```
\begin{tasks}(2)
 \task \sample
 \task \s\ \s
 \task \s
 \task \sample
 \task \s\par\s
\end{tasks}
```

a) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

b) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

c) This is some sample text we will use to create a somewhat longer text spanning a few lines.

d) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

e) This is some sample text we will use to create a somewhat longer text spanning a few lines.

   This is some sample text we will use to create a somewhat longer text spanning a few lines.

21.2 Introducing a New Row

Introduced in version 0.9

Sometimes it may come in handy if the current row of items could be terminated and a new one is started. This is possible with the following command:

► `\startnewitemline`

```
\begin{tasks}(4)
 \task the first
 \task the second
 \task the third
 \task the fourth
 \task \rlap{the fifth item is way too long for this so we
  start a new row} \startnewitemline
 \task the sixth
 \task the seventh
 \task \rlap{the eighth item also is too long} \
  startnewitemline
 \task the nineth
 \task the tenth
```

```
\end{tasks}
```

a) the first b) the second c) the third d) the fourth

e) the fifth item is way too long for this so we start a new row

f) the sixth g) the seventh h) the eighth item also is too long

i) the nineth j) the tenth

22 Available Options

The TASKS package has one package option which also is called when you load EXSHEETS with the `load-tasks` option.

▶ `more`
load additional instances for the `tasks` object, details are explained later in section 23.

The environment itself has some more options, namely these:

▶ `style = <instance>` (initially empty)
Choose the instance to be used. Read more on this in section 26.1.

▶ `counter-format = <counter specs>` (initially empty)
Introduced in version 0.9
Sets a custom label. The letters `tsk` are replaced with the task-counter. An optional argument directly following these letters specifies the counter format: `1`: `\arabic`, `a`: `\alph`, `A`: `\Alph`, `r`: `\roman` and `R`: `\Roman`.

▶ `label-format = <code>` (initially empty)
Changed in version 0.9
Can be used to apply a formatting like, *e.g.*, `\bfseries` to the labels.

▶ `label = <code>` (initially empty)
Changed in version 0.9
Overwrite the automatic label to a custom one.

▶ `label-width = <dim>` Default: `1em`
Sets the width of the item labels.

▶ `label-offset = <dim>` Default: `.3333em`
Introduced in version 0.7
Sets the offset, *i.e.*, the distance between label and item.

▶ `label-align = left|right|center` Default: `left`
Introduced in version 0.7
Determines how the labels are aligned within the label-box whose width is set with `label-width`

▶ `before-skip = <skip>` Default: `0pt`
Sets the skip before the list.

▶ `after-skip = <skip>` Default: `0pt`

Sets the skip after the list.

▶ `after-item-skip = <skip>` Default: `1ex plus 1ex minus 1ex`

Introduced in version 0.9

This vertical skip is inserted between rows of items.

▶ `resume = true|false` Default: `false`

The enumeration will resume from a previous `tasks` environment. In order to use this option properly you shouldn't mix different `tasks` environments that both count their items.

These options can also be set using a setup command:

▶ `\settasks{<options>}`

Now the same list as above but with three columns and a different label:

```
\begin{tasks}[counter-format=(tsk[r]),label-width=4ex](3)
 \task \sample
 \task \s\ \s
 \task \s
 \task \sample
 \task \s\par\s
\end{tasks}
```

(i) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

(ii) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

(iii) This is some sample text we will use to create a somewhat longer text spanning a few lines.

(iv) This is some sample text we will use to create a somewhat longer text spanning a few lines. This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

(v) This is some sample text we will use to create a somewhat longer text spanning a few lines.

This is some sample text we will use to create a somewhat longer text spanning a few lines.

Let's use it inside a question:

```
% since settings are local the following ones will be lost
% outside this example; see the appendix how the solution
% appears there
\settasks{counter-format=qu.tsk,label-width=5ex}
\begin{question}[type=exam]{4}
 I have these two tasks for you. Shall we begin?
 \begin{tasks}(2)
  \task The first task: easy!
  \task The second task: even more so!
 \end{tasks}
\end{question}
\begin{solution}[print]
 Now, let's see\ldots\ ah, yes:
 \begin{tasks}
  \task This is the first solution. Told you it was easy.
  \task This is the second solution. And of course you knew that!
 \end{tasks}
\end{solution}
```

Question 57. 4 P.

I have these two tasks for you. Shall we begin?

57.1 The first task: easy! 57.2 The second task: even more so!

Solution 57.

Now, let's see… ah, yes:

57.1 This is the first solution. Told you it was easy.

57.2 This is the second solution. And of course you knew that!

23 Available Instances

When you use the package option `more` of the TASKS package or load EXSHEETS with the `load-tasks` option there are currently three additional instances for the `tasks` object available:

itemize uses `\labelitemi` as labels.

enumerate enumerates the items with 1., 2., …

multiplechoice a – well – 'multiple choice' list.

```
\begin{tasks}[style=itemize](2)
 \task that's just how\ldots
 \task \ldots we expected
\end{tasks}
\begin{tasks}[style=enumerate](2)
 \task that's just how\ldots
 \task \ldots we expected
\end{tasks}
\begin{tasks}[style=multiplechoice](2)
 \task that's just how\ldots
 \task \ldots we expected
\end{tasks}
```

- that's just how…
- …we expected

1. that's just how…
2. …we expected

☐ that's just how…
☐ …we expected

24 Custom Labels

If you want to change a single label inside a list, you can use the optional argument of `\task`. This will temporarily overwrite the default label.

```
\begin{tasks}[style=itemize]
 \task a standard item
 \task another one
 \task[+] a different one
 \task and another one
\end{tasks}
```

- a standard item
- another one
+ a different one
- and another one

25 New Tasks

It is possible to add custom environments that work like the `tasks` environment.

▶ `\NewTasks[<options>]{<name>}[<separator](<cols>)`
Define environment `{<name>}` that uses `<separator>` to introduce a new item. Default for `<separator>` is `\task`, default for `<cols>` is `1`. The `<options>` are the ones described in section 22.

▶ \RenewTasks[<options>]{<name>}[<separator](<cols>)
Renew environment previously defined with \NewTasks.

The tasks environment is defined as follows:

```
\NewTasks{tasks}
```

The separator does not have to be a control sequence:

```
% preamble:
% \usepackage{dingbat}
\NewTasks[label=\footnotesize\leftthumbsup,label-width=1.3em]{done}[*]
\begin{done}
 * First task
 * Second Task
\end{done}
```

👍 First task

👍 Second Task

Although this might seem handy or even nice I strongly advice against using something different than a command sequence. Remember that the items will be split at *every* occurrence of the separator. So in order to use the separator (here for example for a starred variant of a command) within an item it has to be hidden in braces. This is avoided of you use a command sequence which even doesn't have to be defined.

Let's say you want a choices environment that has three columns in its default state. You could do something like this:

```
% preamble:
% \NewTasks[style=multiplechoice]{choices}[\choice](3)
% \newcommand*\correct{\PrintSolutionsTF{\checkedchoicebox}{\choicebox}}
\begin{question}
\begin{choices}
 \choice First choice
 \choice Second choice
 \choice[\correct] Third choice
\end{choices}
\end{question}
\begin{solution}[print]
\begin{choices}
 \choice First choice
 \choice Second choice
 \choice[\correct] Third choice
\end{choices}
\end{solution}
```

Exercise 58.
☐ First choice ☐ Second choice ☐ Third choice

Solution 58.
☐ First choice ☐ Second choice ☒ Third choice

The last example shows you two additional commands:

- `\choicebox` ☐
- `\checkedchoicebox` ☒

26 Styling TASKS

Equivalent to the styling of ExSheets TASKS uses xtemplate to declare additional instances for the lists.

26.1 The tasks Object

The object that's defined by TASKS is the 'tasks' object. This time there are four instances available for the one template (again 'default') that was defined.

26.1.1 Available Options

This section only lists the options that can be used when defining an instance of the 'default' template. The following subsections will give some examples of their usage.

```
\DeclareTemplateInterface{tasks}{default}{3}
 {
   % option        : type      = default
   enumerate       : boolean   = true    ,
   label           : tokenlist           ,
   indent          : length    = 2.5em   ,
   counter-format  : tokenlist = tsk[a]) ,
   label-format    : tokenlist           ,
   label-width     : length    = 1em     ,
   label-offset    : length    = .3333em ,
   after-item-skip : skip      = 1ex plus 1ex minus 1ex
 }
```

26.1.2 Predefined Instances

This is rather brief this time:

```
% ALPHABETIZE: a) b) c)
\DeclareInstance{tasks}{alphabetize}{default}{}
% available when 'load-tasks=true':
% ITEMIZE:
\DeclareInstance{tasks}{itemize}{default}
  {
    enumerate   = false ,
    label-width = 1.125em
  }
% ENUMERATE:
\DeclareInstance{tasks}{enumerate}{default}
  { counter-format = tsk. }
% MULTIPLECHOICE:
\DeclareInstance{tasks}{multiplechoice}{default}
  {
    enumerate = false       ,
    label     = \choicebox  ,
  }
```

Part IV
The TRANSLATIONS package (v0.9a)

27 Motivation

I wrote this package when I was in need for an expandable version of translator's `\translate`. If you don't need that then there is no need whatsoever for this package as translator from the beamer[31] bundle provides all the functionality TRANSLATIONS does (except for: translator's `\translate` isn't expandable, `\GetTranslation` is.).

28 Requirements

TRANSLATIONS requires the etoolbox package.

29 Usage

These are the commands provided by TRANSLATIONS:

► `\DeclareLanguage{<lang>}`
Declare a language that can be used by TRANSLATIONS. If the language already exists it will be silently redefined. This command can only be used in the preamble.

[31] CTAN: beamer

▶ \DeclareLanguageAlias{<lang2>}{<lang1>}
Declares <lang2> to be an alias of <lang1>. If <lang1> doesn't exist yet a warning will be raised and it will be defined. This command can only be used in the preamble.

▶ \DeclareLanguageDialect{<dialect>}{<lang>}
Declares <dialect> to be a dialect of language <lang>. If a translation for <dialect> is provided it is used by the translation macros. If there is none the corresponding translation for <lang> is used instead.

▶ \NewTranslation{<lang>}{<key>}{<translation>}
Defines a translation of key <key> for the language <lang>. An error will be raised if a translation of <key> already exists. This command can only be used in the preamble.

▶ \RenewTranslation{<lang>}{<key>}{<translation>}
Redefines a translation of key <key> for the language <lang>. An error will be raised if no translation of <key> exists. This command can only be used in the preamble.

▶ \DeclareTranslation{<lang>}{<key>}{<translation>}
Defines a translation of key <key> for the language <lang>. No error will be raised if a translation of <key> already exists. This command can only be used in the preamble.

▶ \DeclareTranslationFallback{<key>}{<fallback>}
Defines a fallback translation for key <key> that is used in case no translation of <key> for the currently active language has been provided. No error will be raised if a fallback for <key> already exists. This command can only be used in the preamble.

▷ \GetTranslationFor{<lang>}{<key>}
Fetches and prints the translation of <key> for the language <lang>. This command is expandable.

▷ \GetTranslation{<key>}
Fetches and prints the translation of <key> for the currently active language (as for example set by babel). This command is expandable.

▶ \SaveTranslationFor{<cmd>}{<lang>}{<key>}
Fetches and saves the translation of <key> for the language <lang> in the macro <cmd>.

▶ \SaveTranslation{<cmd>}{<key>}
Fetches and saves the translation of <key> for the currently active language (as for example set by babel) in the macro <cmd>.

▶ \LoadDictionary{<name>}
Loads a file named <name>-<lang>.trsl where <lang> corresponds to the lowercase name of the current language as defined with \DeclareLanguage. This file should contain the translations for the specified language.

▶ \LoadDictionaryFor{<lang>}{<name>}
Loads a file named <name>-<lang>.trsl.

▶ `\DeclareDictTranslation{<key>}{<translation>}`

Quite a number of languages already are defined, either directly or via an alias. So, before you define a language you should take a look in `translations.sty` if the language doesn't already exist.

Here is a small example of usage:

```
% in the preamble:
% \DeclareTranslation{English}{Kueche}{kitchen}
% \DeclareTranslation{German}{Kueche}{K\"uche}
% \DeclareTranslation{Spanish}{Kueche}{cocina}
% \DeclareTranslation{French}{Kueche}{cuisine}

\GetTranslation{Kueche}
\SaveTranslation\kitchen{Kueche}
\SaveTranslationFor\cuisine{french}{Kueche}

\selectlanguage{ngerman}
\GetTranslation{Kueche} \kitchen\ \GetTranslationFor{spanish}{Kueche}
\cuisine
```

kitchen
Küche kitchen cocina cuisine

There is also preliminary support for dialects:

```
% in the preamble:
% \DeclareTranslation{English}{foo}{foo}
% \DeclareTranslation{British}{foo}{bar}
\GetTranslationFor{English}{foo} \\
\GetTranslationFor{British}{foo} \\
\GetTranslationFor{American}{foo}
```

foo
bar
foo

A typical dictionary file should look as follows:

```
% this is file housing-german.trsl
\ProvideDictionaryFor{German}{housing}[<version info>]
\DeclareDictTranslation{kitchen (housing)}{K\"uche}
\DeclareDictTranslation{bathroom (housing)}{Bad}
\DeclareDictTranslation{living room (housing)}{Wohnzimmer}
\DeclareDictTranslation{bedroom (housing)}{Schlafzimmer}
...
\endinput
```

Part V
The CNTFORMATS package (v0.4)

30 Motivation

CNTFORMATS provides a way to format counters with what I will call patterns. This does not in any way effect the usual LaTeX 2_ε way of treating counters and does not use `\the<counter>` nor is it affected by the redefinition of them.

This package is aimed at package or class authors and probably not very useful for document authors.

31 Requirements

CNTFORMATS requires the etoolbox package.

32 Example

A use case typically looks as follows:

```
\ReadCounterPattern{se.sse}
```

32.0

where the key `se` stands for the current value of the `section` counter and `sse` for `subsection`, respectively. `se.sse` is an example for what will be called *pattern*. The keys for the counters can have optional arguments that specify the format:

```
\stepcounter{subsection}
\ReadCounterPattern{se[R](sse[a])}
```

XXXII(a)

`A` stands for `\Alph` and `r` for `\roman`. A complete list can be found in table 2 on page 60. As you can see you can insert arbitrary other tokens in a pattern that won't be changed.

33 Usage

In order to make counters known to CNTFORMATS the following commands are used:

- `\AddCounterPattern*[<module>]{<counter>}{<pattern>}`
  This command will make the (existing) counter `<counter>` known to CNTFORMATS and assign the pattern `<pattern>` to it.

- `\NewCounterPattern*[<module>]{<counter>}{<pattern>}`
  This command will create a new counter `<counter>`, make it known to CNTFORMATS and assign the pattern `<pattern>` to it.

▶ \ReadCounterFrom[<module>]{<counter>}{<internal cmd>}
If you use one of the commands above with the starred version the number for the pattern is not automatically fetched from the internal \c@<counter>. This can now be assigned with \ReadCounterFrom where <internal cmd> is the macro that holds the number.

The commands above can only be used in the document preamble.

After the creation of these pattern markers one wants to be able to use them. There are a number of macros that allow different aspects of usage.

▶ \ReadCounterPattern[<module>]{<pattern>}
Reads, interprets and prints a pattern.

▷ \@cntfmts@parsed@pattern
After \ReadCounterPattern has been used the current pattern interpretation is stored in this macro. The *interpretation* is *not* what is printed. See the examples below for details.

▶ \ReadCounterPatternFrom[<module>]{<macro that holds pattern>}
Reads, interprets and prints a pattern that's stored in a macro.
Otherwise the same as \ReadCounterPattern.

▶ \SaveCounterPattern{<cmd a>}{<cmd b>}{<pattern>}
Saves the <pattern> in <cmd a> and the interpreted pattern in <cmd b>.

▶ \eSaveCounterPattern[<module>]{<cmd a>}{<cmd b>}{<pattern>}
Saves the <pattern> in <cmd a> and the expanded pattern in <cmd b>.

▶ \SaveCounterPatternFrom[<module>]{<cmd a>}{<cmd b>}{<macro that holds pattern>}
Like \SaveCounterPattern but reads the pattern from a macro.

▶ \eSaveCounterPatternFrom[<module>]{<cmd a>}{<cmd b>}{<macro that holds pattern>}
Like \eSaveCounterPattern but reads the pattern from a macro.

The optional argument <module> should be specific for a package, say, so that different patterns for the section for example don't interfer with each other. If you leave the argument the default module cntfmts is used.

The ExSheets packages uses the commands with the module exsheets. You can find the following lines in ExSheets' code:

```
\AddCounterPattern*[exsheets]{section}{se}
\ReadCounterFrom[exsheets]{section}
  \l__exsheets_counter_sec_int

\NewCounterPattern*[exsheets]{question}{qu}
\ReadCounterFrom[exsheets]{question}
  \l__exsheets_counter_qu_int
```

Now let's see a short example that hopefully explains what the macros do:

```
% preamble
% \NewCounterPattern{testa}{ta}
\setcounter{testa}{11}
\ReadCounterPattern{ta}
\ReadCounterPattern{ta[a]} \\
\ttfamily\makeatletter
\meaning\@cntfmts@parsed@pattern

\bigskip
\SaveCounterPattern\tmpa\tmpb{ta[a]}
\meaning\tmpa \\
\meaning\tmpb

\bigskip
\eSaveCounterPattern\tmpa\tmpb{ta[a]}
\meaning\tmpa \\
\meaning\tmpb
```

```
11 k
macro:->{}\csuse {@cntfmts@read@ta@counter}[a]\relax

macro:->ta[a]
macro:->{}\csuse {@cntfmts@read@ta@counter}[a]\relax

macro:->ta[a]
macro:->{}k\relax
```

You can see that somehow additional (empty) groups and a \relax found their way into the interpreted and thus the expanded pattern. This is due to the fact that reading optional arguments expandably isn't easy and must have some safety net.

34 Predefined Patterns and Formats

CNTFORMATS predefines a number of pattern keys. These are listed in table 1.

Table 1: Predefined Patterns for the module cntfmts.

counter	pattern
chapter	ch
section	se
subsection	sse
subsubsection	ssse
paragraph	pg

Table 2: Predefined Formats

key	format
1	\arabic
a	\alph
A	\Alph
r	\roman
R	\Roman

Table 2 lists the predefined formats. If you want you can add own formats.

► \NewPatternFormat{<pattern>}{<format>}

<format> is a number presentation command like \@alph. This command can only be used in the preamble.

```
% preamble
% \usepackage{alphalph}
% \NewPatternFormat{aa}{\alphalph}
\ReadCounterPattern{se[aa]}
```

ah

Part VI
Appendix

A List of all Solutions used in this Manual

Solution 8.
This is the solution to exercise 8!

Solution 9.
See? This solution gets printed where you have put it in the code of your document.

Fancy name 10.
See? This solution gets printed where you have put it and has a fancy name!

Solution 17.
This will be one of a few solutions printed by the following call of \printsolutions.

Solution 18.
The solution to the question with the nice ID.

Solution 19.
Yet another solution.

Solution 25.
To center a short part of the text body one can use the center environment (1 P.). Inside an environment like table one should use \centering (1 P.). For single lines there is also the \centerline command (1 P.).

Solution 26.
For example two volumes from *Computers & Typesetting*: *TEXbook*, *TEX*, *METAFONTbook*, *METAFONT*, *Computer Modern Typefaces*. Each valid answer is worth 1 P.

★ **Solution 27.**
This is what the solution looks like.

Solution 28.
Try to fill in these blanks. All of them are created by using the `\blank` command.

Solution 57.
Now, let's see… ah, yes:

a) This is the first solution. Told you it was easy.

b) This is the second solution. And of course you knew that!

Solution 58.
☐ First choice ☐ Second choice ☒ Third choice

References

[1] Jason Alexander. examdesign. Version 1.1, Mar. 26, 2001. URL: `http://www.ctan.org/pkg/examdesign/`.

[2] Javier Bezos. enumitem. Version 3.5.2, Sept. 28, 2011. URL: `http://www.ctan.org/pkg/enumitem/`.

[3] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. 1st ed. Reading, Mass.: Addison-Wesley, 1994. 528 pp.

[4] Philip Hirschhorn. exam. Version 2.4, May 22, 2011. URL: `http://www.ctan.org/pkg/exam/`.

[5] Dennis Kletzing. multienum. Version n.n. May 19, 2005. URL: `http://www.ctan.org/pkg/multienum/`.

[6] Donald E. Knuth. *Computers & Typesetting*. 5 vols. Reading, Mass.: Addison-Wesley, 1984–1986.

[7] Donald E. Knuth. *Computers & Typesetting*. Vol. A: *The TEXbook*. Reading, Mass.: Addison-Wesley, 1984.

[8] Donald E. Knuth. *Computers & Typesetting*. Vol. B: *TEX: The Program*. Reading, Mass.: Addison-Wesley, 1986.

[9] Donald E. Knuth. *Computers & Typesetting*. Vol. C: *The METAFONTbook*. Reading, Mass.: Addison-Wesley, 1986.

[10] Donald E. Knuth. *Computers & Typesetting*. Vol. D: *METAFONT: The Program*. Reading, Mass.: Addison-Wesley, 1986.

[11] Donald E. Knuth. *Computers & Typesetting*. Vol. E: *Computer Modern Typefaces*. Reading, Mass.: Addison-Wesley, 1986.

[12] Paul Pichaureau. exercise. Version 1.58, May 8, 2012. URL: http://www.ctan.org/pkg/answers/.

[13] Mike Piff, *current maintainer:* Joseph Wright. answers. Version 2.13, Oct. 11, 2010. URL: http://www.ctan.org/pkg/answers/.

[14] Nicola L. C. Talbot. probsoln. Version 3.02, Dec. 10, 2011. URL: http://www.ctan.org/pkg/probsolns/.

Index