

# Exercise bank - manual

Andreas Strauman

March 30, 2018

## Contents

<b>1</b>	<b>Motivation</b>	<b>2</b>
<b>2</b>	<b>Flow/Moderate start</b>	<b>2</b>
2.1	Select . . . . .	2
2.2	Exclude . . . . .	3
2.3	Multiple . . . . .	3
2.4	Mixnmatch . . . . .	4
2.5	Solutions . . . . .	4
<b>3</b>	<b>Reference</b>	<b>4</b>
3.1	Making sets . . . . .	4
3.2	Environments . . . . .	5
3.3	Internationalization . . . . .	6
3.4	Triggers . . . . .	6
3.5	General reference . . . . .	6
3.6	Counters . . . . .	7

## 1 Motivation

So I wanted a package that allowed me to mix-and-match exercises. I couldn't find it, so I made it.

## 2 Flow/Moderate start

I suspect that working with this package will break your current flow. So let's go through it.

This package assumes you put all of your exercises within the folder named **exercises**

exercises/myexercise.tex

```
\begin{intro}
  This introduces our problem
\end{intro}
\begin{problem}
  This is a partproblem 1,
  and will be hidden (just wait, you'll see)
\end{problem}
\begin{problem}
  This is a partproblem 2.
  This will not be hidden, but become part problem a!
\end{problem}
```

Let's build all of them first. In the main file, (the one where you include this package):

main.tex

```
\documentclass{article}
\usepackage{exbank}
\makeset{myExerciseSet}{myexercise}
\begin{document}
  \buildset{myExerciseSet}
\end{document}
```

This builds the entire set, and adds Problem header and partproblem counters ( (1a) and (1b) ) by default.

### 2.1 Select

Now, let's build only the second problem.

main.tex

```
\documentclass{article}
\usepackage{exbank}
\makeset{myExerciseSet}{\select{myexercise}{2}}
\begin{document}
  \buildset{myExerciseSet}
\end{document}
```

This should only build the intro and the one exercise you `\select`ed!  
Now, say you want to hide the intro. Well all you have to do in this case is make the package treat the intro as a problem in regards to what is `\select`ed. Just add the optional argument `[<intro>]` to `\make`. That is switch

```
\makeset{myExerciseSet}{\select{myexercise}{2}}
```

with

```
\makeset[intro]{myExerciseSet}{\select{myexercise}{3}}
```

Notice that there are 3 ‘partproblems’ now since we have to count the intro!

## 2.2 Exclude

But what if you have an exercise with 12 partproblems, and you only want to exclude the 7th partproblem? Well, then `\Exclude` is here to rescue the day for you.

```
\makeset{myExerciseSet}{\exclude{soManyExercises}{7}}
```

Here it’s important to note that the `[intro]` argument would not make the intros disappear. If we wanted to only exclude the intro from our previous example file `exercises/myexercise.tex` we would do

```
\makeset[intro]{myExerciseSet}{\exclude{myexercise}{1}}
```

So we’re excluding the partproblem 1. But that’s the intro when we send the `[intro]` optional argument

## 2.3 Multiple

In `\makeset` you can just separate exercises with commas! Here is an example:  
Let’s say you have two files with exercises. One located in `exercises/circuits/RLC.tex` and one in `exercises/ohm/ohmsGeneralLaw.tex`, and you want to include partproblem 1 through 5 from `RLC.tex` and all of the exercises from `ohmsGeneralLaw.tex`.

```
\makeset{\select{circuits/RLC}{1,...,5}, ohmsGeneralLaw}
```

This will divide it up with problem headers. So that what is in the `RLC.tex`-file will be Problem 1, and `ohmsGeneralLaw.tex` Problem 2.

## 2.4 Mixnmatch

What if you want to make both of them the same exercise? Well, then you pass the `[nohead]` argument to `\makeset`:

```
\makeset[nohead]{\phead, \select{circuits/RLC}{1,...,5}, ohmsGeneralLaw}
```

The `\phead` command makes a problem header. You can pass them as much as you want:

```
\makeset[nohead]{\phead, \select{circuits/RLC}{1,...,5},
                 ohmsGeneralLaw, \phead, someOtherExercise, moreExercises}
```

## 2.5 Solutions

The last thing to cover then is solutions. In your exercise files you just use the solution environment

```
\begin{solution}
Solution goes here
\end{solution}
```

They are hidden by default, so you would have to use `\DisplaySolutions` in your main file to display them.

That covers the basics. Enjoy

! • `\begin{problem}, \end{problem},`  
`\begin{solution}, \end{solution},`  
`\begin{intro}` and `\end{intro}` has to be on their own line without any spaces!

## 3 Reference

### 3.1 Making sets

```
\makeset[⟨intro,nohead⟩]{⟨filable⟩}
```

This command is the one you use to make a set! Later you use `\buildset` to build the sets you make. The `⟨filable⟩` argument is either the name of the file relative to the `\exerciseDir`-path (default its in the root called `exercises`), or you could use the `\select` or `\exclude` to respectively cherry

pick or exclude exercises. (See their docs).

`[<intro>]` this counts the intro environment as a part problem, so that you can `\select` or `\exclude` the intro

`[<nohead>]` prevents the builder from adding a problem header. This is handy if you want to create an exercise that is composed of multiple parts. You can use the `\phead` to insert the problem header where you want it

```
\makeset[nohead]\{\phead, \select{myexercise}\{1,2,3\}\}
```

```
\exclude{\<exerciseFileName>}\{\<Comma separated numbers>}
```

As you can see in the intro section of the documentation, this is for excluding partproblems To be used in `\makeset`<sup>→ P. 4</sup>

```
\select{\<exerciseFileName>}\{\<Comma separated numbers>}
```

As you can see in the intro section of the documentation, this is for cherry picking partproblems To be used in `\makeset`<sup>→ P. 4</sup>

```
\pplabel{\<label>}
```

Labels a partproblem. You can reference to it later using `\ppref{\<label>}`

```
\ppref{\<label>}
```

Reference a partproblem created by `\pplabel{\<label>}`. This prints e.g. 1c)

```
\ppref{\<label>}
```

Reference a partproblem created by `\pplabel{\<label>}`. This prints e.g. 1

## 3.2 Environments

```
\begin{problem}  
  \<environment content>  
\end{problem}
```

Inside the exercises folder, you keep your exercises. Inside there you'd use a problem environment to write your partproblems. It might be a little confusing that you're using `\begin{problem}` instead of `\begin{partproblem}` when you're writing a partproblem, but it's less typing.

! `\end{problem}` has to be on it's own line without any leading spaces!

```
\begin{solution}  
  \<environment content>  
\end{solution}
```

Things inside here is only visible if `\DisplaySolutions`<sup>→ P. 6</sup> are given before `\begin{document}`

! `\end{solution}` has to be on it's own line without any leading spaces!

### `\DisplaySolutions`

Turns on the solutions, so they are shown.

```
\begin{intro}  
  <environment content>  
\end{intro}
```

Sometimes you'd want to introduce your exercises and tell a little bit about it. Maybe have a figure there also. Those things should go inside this environment. This can be treated as a problem in terms of counting. See `\makeset`<sup>→ P. 4</sup> for more info.

! `\end{intro}` has to be on it's own line without any leading spaces!

## 3.3 Internationalization

```
\translateExBank{<Translation key/vals>}
```

This is to translate the text inside the package. As of now the available key/values are

- Problem
- Solution

The Norwegian translation would then be done with

```
\translateExBank{Problem=Oppgave, Solution=Løsning}
```

## 3.4 Triggers

```
\Trigger{<Any Macro>}
```

See `\At`<sup>→ P. 7</sup>

Triggers available:  
*triggers*

## 3.5 General reference

```
\ownLineNoSpacesGotIt
```

This is to annoy the user enough to get his attention about the requirements of the `problem`<sup>→ P. 5</sup>, `solution`<sup>→ P. 5</sup> and `intro` environments. Compilation won't work unless `\end{problem}`, `\end{solution}` and `\end{intro}` are on their own lines and without any spaces. This warning can be removed by doing `\def\ownLineNoSpacesGotIt` before `\usepackage{exbank}`.

```
\setExercisesDir{<directory>}
```

This is the directory, relative to the file you included the package, where the package should be looking for exercises. Default is `exercises`

This package also includes some extra stuff. For example the `\At` and `\Trigger`

`\At{\AnyMacro}`

Here you can send any macro because it isn't evaluated! For example `\At\BeginSomething` is fine and even if `\BeginSomething` is not defined. Also and when using `\Trigger` it just ignores it if it didn't exist. It's pretty similar in function as to `\AtBeginDocument`.

```
\At\BeginSomething{DoSomething}
```

Which is triggered with

```
\Trigger\BeginSomething, this evaluates to DoSomething
```

### 3.6 Counters

`problemcounter`-counter holds the current problem number and `partproblemcounter`-counter holds the current partproblem *number*.