

# probsoln v3.01: creating problem sheets optionally with solutions

Nicola L.C. Talbot

School of Computing Sciences  
University of East Anglia  
Norwich, Norfolk  
NR4 7TJ, United Kingdom.

<http://theoval.cmp.uea.ac.uk/~nlct/>

2011-08-22

## Contents

<b>1 Introduction</b>	<b>2</b>
<b>2 Package Options</b>	<b>2</b>
<b>3 Showing and Hiding Solutions</b>	<b>2</b>
<b>4 General Formatting Commands</b>	<b>3</b>
<b>5 Defining a Problem</b>	<b>5</b>
<b>6 Using a Problem</b>	<b>7</b>
<b>7 Loading Problems From External Files</b>	<b>7</b>
<b>8 Iterating Through Datasets</b>	<b>9</b>
<b>9 Random Number Generator</b>	<b>11</b>
<b>10 Compatibility With Versions Prior to 3.0</b>	<b>12</b>
<b>11 The Code</b>	<b>13</b>
11.1 Package Definition . . . . .	13
11.2 Package Options . . . . .	14
11.3 Databases . . . . .	15
11.4 Defining New Problems . . . . .	17
11.5 Using Problems . . . . .	21
11.6 Loading Problems From Another File . . . . .	22
11.7 Iterating Through a Data Base . . . . .	25
11.8 Random Numbers . . . . .	26
11.9 Compatibility With Older Versions . . . . .	29

11.10 Formatting Commands . . . . .	30
-------------------------------------	----

<b>Index</b>	<b>32</b>
--------------	-----------

## 1 Introduction

The `probsoln` package is designed for teachers or lecturers who want to create problem sheets for their students. This package was designed with mathematics problems in mind, but can be used for other subjects as well. The idea is to create a file containing a large number of problems with their solutions which can be read in by  $\text{\LaTeX}$ , and then select a number of problems to typeset. This means that once the database has been set up, each year you can easily create a new problem sheet that is sufficiently different from the previous year, thus preventing the temptation of current students seeking out the previous year's students, and checking out their answers. There is also an option that can be passed to the package to determine whether or not the solutions should be printed. In this way, one file can either produce the student's version or the teacher's version.

## 2 Package Options

The following options may be passed to this package:

- answers** Show the answers
- noanswers** Don't show the answers (default)
- draft** Display the label and dataset name when a problem is used
- final** Don't display label and dataset name when a problem is used
- usedefaultargs** Make `\thisproblem` use the default arguments supplied in the problem definition.
- nousedefaultargs** Make `\thisproblem` prompt for problem arguments (default).

## 3 Showing and Hiding Solutions

In addition to the `answers` and `noanswers` package options, it is also possible to show or suppress the solutions using

<code>\showanswers</code>	<code>\showanswers</code>
and	
<code>\hideanswers</code>	<code>\hideanswers</code>

respectively.

The boolean variable `showanswers` determines whether the answers should be displayed. You can use this value with the `ifthen` package to specify different text depending on whether the solutions should be displayed. For example:

```
Assignment 1\ifthenelse{\boolean{showanswers}}{(Solution Sheet)}{}
```

Alternatively you can use `\ifshowanswers... \else... \fi`:

```
Assignment 1\ifshowanswers\space (Solution Sheet)\fi
```

`onlyproblem` For longer passages, you can use the environments `onlyproblem` and `onlysolution`.  
`onlysolution` For example:

```
\begin{onlyproblem}%  
What is the derivative of  $f(x) = x^2$ ?  
\end{onlyproblem}%  
\begin{onlysolution}%  
 $f'(x) = 2x$   
\end{onlysolution}
```

The above will only display the question if `showanswers` is false and will only display the solution if `showanswers` is true. If you want the question to appear in the answer sheet as well as the solution, then don't put the question in the `onlyproblem` environment:

```
What is the derivative of  $f(x) = x^2$ ?  
\begin{onlysolution}%  
Solution:  $f'(x) = 2x$   
\end{onlysolution}
```

You can't use verbatim text in the body of the `onlyproblem` or `onlysolution` environments. If you need verbatim-like text, then try packages such as `alltt`. Remember that you also can't use verbatim text in command arguments.

If you use `onlysolution` within the `defproblem` environment, the problem will be tagged as having a solution and will be added to the list used by `\foreachsolution`.

## 4 General Formatting Commands

The commands and environments described in this section are provided to assist formatting problems and their solutions.

`solution` `\begin{solution}<text>\end{solution}`

By default, this is equivalent to

```
\par\noindent\textbf{\solutionname}: <text>
```

`\solutionname` where `\solutionname` defaults to "Solution". Note that you must place the `solution` environment inside the `onlysolution` environment or between `\ifshowanswers... \fi` to ensure that it is suppressed when the solutions are not wanted. (See Section 3.)

Note that the `probsoln` package will only define the `solution` environment if it is not already defined.

`textenum` `\begin{textenum}... \end{textenum}`

The `textenum` environment is like the `enumerate` environment but is in-line. It uses the same counter that the `enumerate` environment would use at that level so the question can be compact but the answer can use `enumerate` instead. For example:

```
\begin{onlyproblem}%
  Differentiate the following:
  \begin{textenum}
    \item  $f(x)=2^x$ ; \item  $f(x)=\cot(x)$ 
  \end{textenum}
\end{onlyproblem}
\begin{onlysolution}
  \begin{enumerate}
    \item
      \begin{align*}
        f(x) &= 2^x = \exp(\ln(x^2)) = \exp(2\ln(x))\\
        f'(x) &= \exp(2\ln(x)) \times \frac{2}{x} \\
        &= f(x) \frac{2}{x}
      \end{align*}
    \item
      \begin{align*}
        f(x) &= \cot(x) = (\tan(x))^{-2} \\
        f'(x) &= -(\tan(x))^{-2} \times \sec^2(x) \\
        &= -\csc^2x
      \end{align*}
    \end{enumerate}
  \end{onlysolution}
```

In this example, the items in the question are brief, so an `enumerate` environment would result in a lot of unnecessary white space, but the answers require more space, so an `enumerate` environment is more appropriate. Since the `textenum` environment uses the same counters as the `enumerate` environment, the question and answer sheets use consistent labelling. Note that there are other packages available on CTAN that you can use to create in-line lists. Check the TeX Catalogue<sup>1</sup> for further details.

`\correctitem`  
`\incorrectitem`

```
\correctitem
\incorrectitem
```

You can use the commands `\correctitem` and `\incorrectitem` in place of `\item`. If the solutions are suppressed, these commands behave in the same way as `\item`, otherwise they format the item label using one of the commands:

`\correctitemformat`  
`\incorrectitemformat`

```
\correctitemformat{\label}
\incorrectitemformat{\label}
```

For example:

Under which of the following functions does  $S=\{a_1, a_2\}$  become a probability space?  
`\begin{enumerate}`

<sup>1</sup><http://www.tex.ac.uk/tex-archive/help/Catalogue/bytopic.html#enumeration>

```

\incorrectitem $P(a_1)=\frac{1}{3}$, $P(a_2)=\frac{1}{2}$
\correctitem $P(a_1)=\frac{3}{4}$, $P(a_2)=\frac{1}{4}$
\correctitem $P(a_1)=1$, $P(a_2)=0$
\incorrectitem $P(a_1)=\frac{5}{4}$, $P(a_2)=-\frac{1}{4}$
\end{enumerate}

```

The default definition of `\correctitemformat` puts a frame around the label.

## 5 Defining a Problem

It is possible to construct a problem sheet with solutions using the commands described in the previous sections, however it is also possible to define a set of problems for later use. In this way you can create an external file containing many problems some or all of which can be loaded and used in a document. The `probsoln` package has a default data set labelled “default” in which you can store problems. Alternatively, you can create multiple data sets. You can then iterate through each problem in a problem set. You can use a previously defined problem more than once, which means that by judicious use of `onlyproblem`, `onlysolution` or the `showanswers` boolean variable in conjunction with `\showanswers` and `\hideanswers`, you can print the solutions in a different location to the questions (for example in an appendix).

defproblem

```

\begin{defproblem}[\langle n \rangle][\langle default args \rangle]{\langle label \rangle}
\langle definition \rangle
\end{defproblem}

```

This defines the problem whose label is given by  $\langle label \rangle$ . The label must be unique for a given data set and should not contain active characters or a comma. (Active characters include the special characters such as  $\$$  and  $\&$ , but some packages may make other symbols active, such as the colon ( $:$ ) character. For example, the `ngerman` and `babel` packages make certain punctuation active. Check the relevant package documentation for details.)

If `defproblem` occurs in the document or is included via `\input` or `\include`, then the problem will be added to the default data set. If `defproblem` occurs in an external file that is loaded using one of the commands defined in Section 7 then the problem will be added to the specified data set.

The contents of the `defproblem` environment should be the text that defines the problem. This may include any of the commands defined in Section 3 and Section 4.

The problem may optionally take  $\langle n \rangle$  arguments (where  $\langle n \rangle$  is from 0 to 9). The arguments can be referenced in the definition via  $\#1, \dots, \#9$ . If  $\langle n \rangle$  is omitted then the problem doesn’t take any arguments. The following example defines a problem with one argument:

```

\begin{defproblem}[1]{diffsin}
Differentiate $f(x)=\sin(\#1x)$.
\begin{onlysolution}%
\begin{solution}
$f'(x) = \#1\cos(\#1x)$
\end{solution}

```

```
\end{onlysolution}
\end{defproblem}
```

The second optional argument  $\langle default\ args \rangle$  supplies default problem arguments that will automatically be used within `\thisproblem` when used in `\foreachproblem` in conjunction with the package option `usedefaultargs`. (See Section 8.) For example:

```
\begin{defproblem}[1][2]{diffsin}
Differentiate  $f(x)=\sin(\#1x)$ .
\begin{onlysolution}%
  \begin{solution}
     $f'(x) = \#1\cos(\#1x)$ 
  \end{solution}
\end{onlysolution}
\end{defproblem}
```

`\newproblem`

```
\newproblem[ $\langle n \rangle$ ][ $\langle default\ args \rangle$ ]{ $\langle label \rangle$ }{ $\langle problem \rangle$ }{ $\langle solution \rangle$ }
```

This is a shortcut command for:

```
\begin{defproblem}[ $\langle n \rangle$ ][ $\langle default\ args \rangle$ ]{ $\langle label \rangle$ }%
 $\langle problem \rangle$ %
\begin{onlysolution}%
\begin{solution}%
 $\langle solution \rangle$ %
\end{solution}%
\end{onlysolution}%
\end{defproblem}
```

For example:

```
\newproblem[1]{diffsin}{%
  \f(x) = \sin(\#1x)\}
}%
{%
  \f'(x) = \#1\cos(\#1x)\}
}
```

is equivalent to

```
\begin{defproblem}[1]{diffcos}%
  \f(x) = \cos(\#1x)\}
\begin{onlysolution}%
  \begin{solution}%
    \f'(x) = -\#1\sin(\#1x)\}
  \end{solution}%
\end{onlysolution}%
\end{defproblem}
```

(In this example, the argument will need to be a positive number to avoid a double minus in the answer. If you want to perform floating point arithmetic on the arguments, then try the `fp` or `pgfmath` packages.)

Alternatively, if you want to supply default arguments to use when iterating through problems with `\foreachproblem`:

```

\newproblem[1][3]{diffsin}{%
  \f(x) = \sin(#1x)\}
}%
{%
  \f'(x) = #1\cos(#1x)\}
}

```

```

\newproblem* \newproblem*[\langle n \rangle][\langle default args \rangle]{\langle label \rangle}{\langle definition \rangle}

```

This is a shortcut for:

```

\begin{defproblem}[\langle n \rangle][\langle default args \rangle]{\langle label \rangle}%
  \langle definition \rangle%
\end{defproblem}

```

Verbatim text must not be used in the contents of `defproblem` or in any of the arguments to `\newproblem` or `\newproblem*`. If you need verbatim-like text consider using `\textttt` or the `alltt` package.

## 6 Using a Problem

Once you have defined a problem using `defproblem` or `\newproblem` (see Section 5), you can later display the problem using:

```

\useproblem \useproblem[\langle data set \rangle]{\langle label \rangle}{\langle arg_1 \rangle} \dots {\langle arg_N \rangle}

```

where `\langle data set \rangle` is the name of the data set that contains the problem (the default data set is used if omitted), `\langle label \rangle` is the label identifying the required problem and `\langle arg1 \rangle`, ..., `\langle argN \rangle` are the arguments to pass to the problem, if the problem was defined to have arguments (where  $N$  is the number of arguments specified when the problem was defined).

For example, in the previous section the problem `diffcos` was defined to have one argument, so it can be used as follows:

```

\useproblem{diffcos}{3}

```

This will be equivalent to:

```

\f(x) = \cos(3x)\}
\begin{onlysolution}%
\begin{solution}%
  \f'(x) = -3\sin(3x)\}
\end{solution}%
\end{onlysolution}%

```

## 7 Loading Problems From External Files

You can store all your problem definitions (see Section 5) in an external file. These problems can all be appended to the default data set by including the file

via `\input` or they can be appended to other data sets using one of the commands described below. Once you have loaded all the required problems, you can iterate through the data sets using the commands described in Section 8. Note that the commands below will create a new data set, if the named data set doesn't exist.

`\loadallproblems`      `\loadallproblems[ $\langle data set \rangle$ ]{ $\langle filename \rangle$ }`

This will load all problems defined in  $\langle filename \rangle$  and append them to the specified data set, in the order in which they are defined in the file. If  $\langle data set \rangle$  is omitted, the default data set will be used. If  $\langle data set \rangle$  doesn't exist, it will be created.

`\loadselectedproblems`      `\loadselectedproblems[ $\langle data set \rangle$ ]{ $\langle labels \rangle$ }{ $\langle filename \rangle$ }`

This is like `\loadallproblems`, but only those problems whose label is listed in the comma-separated list  $\langle labels \rangle$  are loaded. For example, if I have some problems defined in the file `derivatives.tex`, then

```
\loadselectedproblems{diffsin,diffcos}{derivatives}
```

will only load the problems whose labels are `diffsin` and `diffcos`, respectively. All the other problems in the file will remain undefined.

`\loadexceptproblems`      `\loadexceptproblems[ $\langle data set \rangle$ ]{ $\langle exception list \rangle$ }{ $\langle filename \rangle$ }`

This is the reverse of `\loadselectedproblems`. This loads all problems except those whose labels are listed in  $\langle exception list \rangle$ .

`\loadrandomproblems`      `\loadrandomproblems[ $\langle data set \rangle$ ]{ $\langle n \rangle$ }{ $\langle filename \rangle$ }`

This randomly loads  $\langle n \rangle$  problems from  $\langle filename \rangle$  and adds them to the given data set. If  $\langle data set \rangle$  is omitted, the default data set is assumed. Note that the problems will be added to the data set in a random order, not in the order in which they were defined. There must be at least  $\langle n \rangle$  problems defined in  $\langle filename \rangle$ .

`\loadrandomexcept`      `\loadrandomexcept[ $\langle data set \rangle$ ]{ $\langle n \rangle$ }{ $\langle filename \rangle$ }{ $\langle exception list \rangle$ }`

This is similar to `\loadrandomproblems` except that it won't load those problems whose labels are listed in  $\langle exception list \rangle$ .

Note that the random number generator has been modified in version 3.01 in order to fix a bug. If you want to ensure that your random numbers are compatible with earlier versions, you can switch to the old generator using

`\PSNuseoldrandom`      `\PSNuseoldrandom`



It is generally not a good idea to place anything in  $\langle filename \rangle$  that is not inside the body of `defproblem` or in the arguments to `\newproblem` or `\newproblem*`. All the commands in this section input the external file within a local scope, so command definitions would need to be made global to have any effect. In addition, `\loadrandomproblems` has to load each file twice, which means that anything outside a problem definition will be parsed twice.

## 8 Iterating Through Datasets

Once you have defined all your problems for a given data set, you can use an individual problem with `\useproblem` (see Section 6) but it is more likely that you will want to iterate through all the problems so that you don't need to remember the labels of all the problems you have defined.

`\foreachproblem` `\foreachproblem[\langle data set \rangle]{\langle body \rangle}`

This does  $\langle body \rangle$  for each problem in the given data set. If  $\langle data set \rangle$  is omitted, the default data set is used. Within  $\langle body \rangle$  you can use

`\thisproblem` `\thisproblem`

to use the current problem and

`\thisproblemlabel` `\thisproblemlabel`

to access the current label. If the problem requires arguments, and no default arguments were supplied in the problem definition or the package option `usedefaultargs` was not used, then you will be prompted for arguments, so if you want to use this approach you will need to use L<sup>A</sup>T<sub>E</sub>X in interactive mode. If you do provide arguments, they will be stored in the event that you need to iterate through the data set again. The arguments will be included in `\thisproblem`, so you only need to use `\thisproblem` without having to specify `\useproblem`.

For example, to iterate through all problems in the default data set:

```
\begin{enumerate}
\foreachproblem{\item\thisproblem}
\end{enumerate}
```

`\foreachsolution` `\foreachsolution[\langle data set \rangle]{\langle body \rangle}`

This is equivalent to `\foreachproblem`, but only iterates through problems that contain the `onlvsolution` environment. Note that you still need to use `\showanswers` or the `answers` package option for the contents of the `onlvsolution` environment to appear.

`\foreachdataset` `\foreachdataset{\langle cmd \rangle}{\langle body \rangle}`

This does  $\langle body \rangle$  for each of the defined data sets. Within  $\langle body \rangle$ ,  $\langle cmd \rangle$  will be set to the name of the current data set. For example, to display all problems in all data sets:

```
\begin{enumerate}
\foreachdataset{\thisdataset}{%
\foreachproblem[\thisdataset]{\item\thisproblem}}
\end{enumerate}
```

Suppose I have two external files called `derivatives.tex` and `probspaces.tex` which define problems using both `onlyproblem` and `onlysolution` for example:

```
\begin{defproblem}{cosxsqsinx}%
\begin{onlyproblem}%
$y = \cos(x^2)\sin x$.%
\end{onlyproblem}%
\begin{onlysolution}%

$$\left[\frac{dy}{dx} = -\sin(x^2)2x\sin x + \cos(x^2)\cos x\right]$$

\end{onlysolution}%
\end{defproblem}
```

I can write a document that creates two data sets, one for the derivative problems and one for the problems about probability spaces. I can then use `\hideanswers` and iterate through the require data set to produce the problems. Later, I can use `\showanswers` and iterate over all problems defined in both data sets to produce the chapter containing all the answers. When displaying the questions, I have taken advantage of the fact that I can cross-reference items within an `enumerate` environment, and redefined `\theenumi` to label the questions according to the chapter. The cross-reference label is constructed from the problem label and is referenced in the answer section to ensure that the answers have the same label as the questions.

```
\documentclass{report}
\usepackage{probsoln}
\begin{document}
\hideanswers
\chapter{Differentiation}
% randomly select 25 problems from derivatives.tex and add to
% the data set called 'deriv'
\loadrandomproblems[deriv]{25}{derivatives}

% Display the problems
\renewcommand{\theenumi}{\thechapter.\arabic{enumi}}
\begin{enumerate}
\foreachproblem[deriv]{\item\label{prob:\thisproblemlabel}\thisproblem}
\end{enumerate}
% You may need to change \theenumi back here

\chapter{Probability Spaces}
% randomly select 25 problems from probspaces.tex and add to
% the data set called 'spaces'
\loadrandomproblems[spaces]{25}{probspaces}

% Display the problems
\renewcommand{\theenumi}{\thechapter.\arabic{enumi}}
```

```

\begin{enumerate}
\foreachproblem[spaces]{\item\label{prob:\thisproblemlabel}\thisproblem}
\end{enumerate}
% You may need to change \thenumi back here

\appendix

\chapter{Solutions}
\showanswers
\begin{itemize}
\foreachdataset{\thisdataset}{%
\foreachproblem[\thisdataset]{\item[\ref{prob:\thisproblemlabel}]\thisproblem}
}
\end{itemize}

\end{document}

```

## 9 Random Number Generator

This package provides a pseudo-random number generator that is used by `\loadrandomproblems`. As noted earlier the random number generator has been modified in version 3.01 in order to fix a bug. If you want to ensure that your random numbers are compatible with earlier versions, you can switch to the old generator using

`\PSNuseoldrandom`

```
\PSNuseoldrandom
```

`\PSNrandseed`

```
\PSNrandseed{\langle n \rangle}
```

This sets the seed to  $\langle n \rangle$  which must be a non-zero integer. For example, to generate a different set of random numbers every time you L<sup>A</sup>T<sub>E</sub>X your document,<sup>2</sup> put the following in your preamble:

```
\PSNrandseed{\time}
```

or to generate a different set of random numbers every year you L<sup>A</sup>T<sub>E</sub>X your document:

```
\PSNrandseed{\year}
```

`\PSNgetrandseed`

```
\PSNgetrandseed{\langle register \rangle}
```

This stores the current seed in the count register specified by  $\langle register \rangle$ . For example:

```
\newcount\myseed
\PSNgetrandseed{\myseed}
```

`\PSNrandom`

```
\PSNrandom{\langle register \rangle}{\langle n \rangle}
```

<sup>2</sup>assuming you leave at least a minute between runs.

Generates a random integer from 1 to  $\langle n \rangle$  and stores in the count register specified by  $\langle register \rangle$ . For example, the following generates an integer from 1 to 10 and stores it in the register `\myreg`:

```
\newcount\myreg
\PSNrandom{\myreg}{10}
```

`\random` `\random{\langle counter \rangle}{\langle min \rangle}{\langle max \rangle}`

Generates a random integer from  $\langle min \rangle$  to  $\langle max \rangle$  and stores in the given counter. For example, the following generates a random number between 3 and 8 (inclusive) and stores it in the counter `myrand`.

```
\newcounter{myrand}
\random{myrand}{3}{8}
```

`\doforrandN` `\doforrandN{\langle n \rangle}{\langle cmd \rangle}{\langle list \rangle}{\langle text \rangle}`

Randomly selects  $\langle n \rangle$  values from the comma-separated list given by  $\langle list \rangle$  and iterates through this subset. On each iteration it sets  $\langle cmd \rangle$  to the current value and does  $\langle text \rangle$ . For example, the following will load a randomly selected problem from two of the listed files (where `file1.tex`, `file2.tex` and `file3.tex` are files containing at least one problem):

```
\doforrandN{2}{\thisfile}{file1,file2,file3}{%
\loadrandomproblems{1}{\thisfile}}
```

## 10 Compatibility With Versions Prior to 3.0

Version 3.0 of the `probsoln` package completely changed the structure of the package, but the commands described in this section have been provided to maintain compatibility with earlier versions. The only problems that are likely to occur are those where commands are contained within groups. This will effect any commands that are contained in external files that are outside of the arguments to `\newproblem` and `\newproblem*`. However, since the external files had to be parsed twice in order to load the problems, this shouldn't be an issue as adding anything other than problem definitions in those files would be problematic anyway.

The other likely difference is where the random generator is used in a group. This includes commands such as `\selectrandomly`. For example, if your document contained something like:

```
\begin{enumerate}
\selectrandomly{file1}{8}

\item Solve the following:
\begin{enumerate}
\selectrandomly{file2}{4}
\end{enumerate}
\end{enumerate}
```

```
\selectrandomly{file3}{2}
\end{enumerate}
```

Then using versions prior to v3.0 will produce a different set of random numbers since the second `\selectrandomly` is in a different level of grouping. If you want to ensure that the document produces exactly the same random set with the new version as with the old version, you will need to get and set the random number seed. For example, the above would need to be modified so that it becomes:

```
\begin{enumerate}
\selectrandomly{file1}{8}

\item Solve the following:
\newcount\oldseed
\PSNgetrandseed{\oldseed}
\begin{enumerate}
\selectrandomly{file2}{4}
\end{enumerate}
\PSNrandseed{\oldseed}

\selectrandomly{file3}{2}
\end{enumerate}
```

`\selectrandomly`

```
\selectrandomly{<filename>}{<n>}
```

This is now equivalent to:

```
{\loadrandomproblems[<filename>]{<n>}{<filename>}}%
\foreachproblem[<filename>]{\PSNitem\thisproblem\endPSNitem}
```

`\selectallproblems`

```
\selectallproblems{<filename>}
```

This is now equivalent to:

```
{\loadallproblems[<filename>]{<filename>}}%
\foreachproblem[<filename>]{\PSNitem\thisproblem\endPSNitem}
```

Note that in both the above cases, a new data set is created with the same name as the file name.

## 11 The Code

### 11.1 Package Definition

This package requires L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

```
\NeedsTeXFormat{LaTeX2e}
```

Identify this package and version:

```
\ProvidesPackage{probsoln}[2011/08/22 v3.01 (NLCT)]
```

Required packages:

```
\RequirePackage{ifthen}
\RequirePackage{amsmath}
```

## 11.2 Package Options

`\ifshowanswers` Define boolean to determine whether or not to show the solutions. This governs whether the contents of `onlysolution` and `onlyproblem` are displayed.

```
\newif\ifshowanswers
\showanswersfalse
```

`\showanswers` Define synonym for `\showanswerstrue`

```
\let\showanswers\showanswerstrue
```

`\hideanswers` Define synonym for `\showanswersfalse`

```
\let\hideanswers\showanswersfalse
```

The package option `answers` displays the solutions.

```
\DeclareOption{answers}{\showanswerstrue}
```

The package option `noanswers` hides the solutions.

```
\DeclareOption{noanswers}{\showanswersfalse}
```

Determine whether or not to use default arguments for problems.

`\ifusedefaultprobargs`

```
\newif\ifusedefaultprobargs
```

`usedefaultargs`

```
\DeclareOption{usedefaultargs}{\usedefaultprobargstrue}
```

`usedefaultargs`

```
\DeclareOption{nousedefaultargs}{\usedefaultprobargsfalse}
\usedefaultprobargsfalse
```

`\prob@showdraftlabel` `\prob@showdraftlabel{<db name>}{<label>}`

Used by `\useproblem` to display data base name and problem label when in draft mode.

```
\newcommand*\prob@showdraftlabel}[2]{}
```

`\draftproblemlabel` `\draftproblemlabel{<db name>}{<label>}`

Displays the data base name and label.

```
\newcommand*\draftproblemlabel}[2]{[#1,#2]}
```

Draft mode displays the problem label using `\draftproblemlabel`

```
\DeclareOption{draft}{%
```

```
\renewcommand*\prob@showdraftlabel}[2]{\draftproblemlabel{#1}{#2}}
```

Final mode:

```
\DeclareOption{final}{%
```

```
\renewcommand*\prob@showdraftlabel}[2]{}
```

Process package options:

```
\ProcessOptions
```

## 11.3 Databases

All the problems are stored in data bases. Each data base  $\langle name \rangle$  is represented as a macro  $\backslash\text{prob@db@}\langle name \rangle$  which stores a comma-separated list of labels for each problem associated with that data base. Each problem  $\langle label \rangle$  is stored in the macro  $\backslash\text{prob@data@}\langle name \rangle@ \langle name \rangle@ \langle label \rangle$ . Problems loaded from an external file using  $\backslash\text{loadproblems}$  are added to the specified data base. Any problems that are defined in the document or are  $\backslash\text{inputed}$  from another file (without the use of  $\backslash\text{loadproblems}$ ) are added to the default data base.

Define the default data base:

```
\newcommand*\prob@db@default{}
```

$\backslash\text{prob@databases}$  Store a list of all the defined data bases.

```
\newcommand*\prob@databases{default}
```

Each defined database has a list of undisplayed solutions.

```
\newcommand*\prob@db@default@solutions{}
```

$\backslash\text{prob@newdb}$   $\backslash\text{prob@newdb}\{\langle name \rangle\}$

Creates a new (empty) data base.

```
\newcommand*\prob@newdb}[1]{%
\@ifundefined{prob@db@#1}{%
\expandafter\gdef\csname prob@db@#1\endcsname{%
\xdef\prob@databases{\prob@databases,#1}%
\expandafter\gdef\csname prob@db@#1@solutions\endcsname{%
}%
\PackageError{probsoln}{Data set ‘#1’ is already defined}{}}
```

$\backslash\text{prob@currentdb}$  Keep a track of the current data base

```
\newcommand*\prob@currentdb{default}
```

$\backslash\text{moveproblem}$   $\backslash\text{moveproblem}\{\langle label \rangle\}\{\langle source \rangle\}\{\langle target \rangle\}$

```
\newcommand{\moveproblem}[3]{%
\@moveproblem{#1}{#2}{#3}%
\expandafter\let\expandafter\@tmpdblist
\csname prob@db@#2\endcsname
\expandafter\gdef\csname prob@db@#2\endcsname{%
\@for\@tmplab:=\@tmpdblist\do{%
\ifthenelse{\equal{\@tmplab}{#1}}{ }{%
\expandafter\ifx\csname prob@db@#2\endcsname\@empty
\expandafter\xdef\csname prob@db@#2\endcsname{\@tmplab}%
\else
\expandafter\xdef\csname prob@db@#2\endcsname{%
\csname prob@db@#2\endcsname,%
\@tmplab}%
\fi
}%
}
```

```
}%
}
```

```
\@moveproblem \@moveproblem{<label>}{<source>}{<target>}
```

Moves problem identified by  $\langle label \rangle$  from the data base  $\langle source \rangle$  to the data base  $\langle target \rangle$ . (Doesn't remove label from  $\langle source \rangle$  — that needs to be done separately.)

```
\newcommand*{\@moveproblem}[3]{%
```

Add label to target data base

```
\expandafter\ifx\csname prob@db@#3\endcsname\@empty
\expandafter\xdef\csname prob@db@#3\endcsname{#1}%
\else
\expandafter\xdef\csname prob@db@#3\endcsname{%
\csname prob@db@#3\endcsname,#1}%
\fi
```

Redefine  $\backslash prob@data@<source>@<label>$  as  $\backslash prob@data@<target>@<label>$ .

```
\edef\do@movedata{%
\noexpand\global\noexpand\let\expandafter\noexpand
\csname prob@data@#3@#1\endcsname=%
\expandafter\noexpand\csname prob@data@#2@#1\endcsname
\noexpand\global\noexpand\let
\expandafter\noexpand
\csname prob@data@#2@#1\endcsname=\noexpand\undefined
}%
\do@movedata
```

Redefine  $\backslash prob@argN@<source>@<label>$  as  $\backslash prob@argN@<target>@<label>$ .

```
\edef\do@moveargN{%
\noexpand\global\noexpand\let\expandafter\noexpand
\csname prob@argN@#3@#1\endcsname=%
\expandafter\noexpand\csname prob@argN@#2@#1\endcsname
\noexpand\global\noexpand\let
\expandafter\noexpand
\csname prob@argN@#2@#1\endcsname=\noexpand\undefined
}%
\do@moveargN
```

Redefine  $\backslash prob@args@<source>@<label>$  as  $\backslash prob@args@<target>@<label>$ , if defined.

```
\@ifundefined{prob@args@#2@#1}{-}{%
\edef\do@moveargs{%
\noexpand\global\noexpand\let\expandafter\noexpand
\csname prob@args@#3@#1\endcsname=%
\expandafter\noexpand\csname prob@args@#2@#1\endcsname
\noexpand\global\noexpand\let
\expandafter\noexpand
\csname prob@args@#2@#1\endcsname=\noexpand\undefined
}%
\do@moveargs
}%
}
```



## 11.4 Defining New Problems

`\prob@newproblem` `\prob@newproblem{<n>}{<db name>}{<label>}{<definition>}{<default args>}`

Define a new problem identified by `<label>` for data base `<db name>` with the given definition. The problem has `<n>` arguments (each represented by #1 etc). The arguments may either be appended to `\useproblem{<label>}` or may be stored in `\prob@args@<db name>@<label>` if the problem is to be accessed via `\foreachproblem`. The number of arguments is stored in `\prob@argN@<db name>@<label>`

```
\newcommand{\prob@newproblem}[5]{%
```

If the given data base is not defined, create it:

```
\@ifundefined{prob@db@#2}{\prob@newdb{#2}}{%
```

Check whether this entry has already been defined:

```
\@ifundefined{prob@data@#2@#3}%
{%
```

Define the new problem and make it global:

```
\let\@tmp=\undefined
\newcommand\@tmp[#1]{#4}%
\expandafter\global\expandafter\let
\csname prob@data@#2@#3\endcsname=\@tmp
\expandafter\xdef\csname prob@argN@#2@#3\endcsname{\number#1}%
\let\@tmp=\undefined
```

Add the label to the data base:

```
\expandafter\ifx\csname prob@db@#2\endcsname\@empty
\expandafter\xdef\csname prob@db@#2\endcsname{#3}%
\else
\expandafter\xdef\csname prob@db@#2\endcsname{%
\csname prob@db@#2\endcsname,#3}%
\fi
```

If default arguments supplied, set them

```
\ifx\@empty#5\@empty
\else
\edef\thisprob@currentdb{#2}%
\edef\thisprob@currentlabel{#3}%
\expandafter\@setdefaultprobargs\expandafter{#5}%
\fi
}%
}%
\PackageError{probsoln}{Problem '#3' is already defined for
data base '#2'}{Problem labels must be unique for each data base}%
}%
}
```

`\@prob@gobblethree` Ignore all three arguments

```
\newcommand{\@prob@gobblethree}[3]{}
```

`\@prob@getargs` `\@prob@getargs<n><db name><label>`

Prompt user for  $\langle n \rangle$  arguments for problem  $\langle label \rangle$  in data base  $\langle db name \rangle$ .

```
\newcommand{\@prob@getargs}[3]{%
\message{Problem '#3' (in data base '#2') requires #1 argument(s).^^J
Please specify (e.g. \csname @prob@getargs@eg@romannumeral#1\endcsname):}%
\read-1to\@tmp
\expandafter\global\expandafter\let
\csname prob@args@#2@#3\endcsname=\@tmp
}
\def\@prob@getargs@eg@i{{12}}
\def\@prob@getargs@eg@ii{{5}{3}}
\def\@prob@getargs@eg@iii{{4}{5}{2}}
\def\@prob@getargs@eg@iv{{5}{3}{10}{8}}
\def\@prob@getargs@eg@v{{5}{3}{10}{8}{4}}
\def\@prob@getargs@eg@vi{{5}{3}{10}{8}{4}{24}}
\def\@prob@getargs@eg@vii{{5}{3}{10}{8}{4}{24}{32}}
\def\@prob@getargs@eg@viii{{5}{3}{10}{8}{4}{24}{32}{9}}
\def\@prob@getargs@eg@ix{{5}{3}{10}{8}{4}{24}{32}{9}{2}}
```

`\@prob@do@getargs`

```
\let\@prob@do@getargs\@prob@gobblethree
```

`\setprobargs` `\setprobargs[\langle db name \rangle]{\langle label \rangle}{\langle args \rangle}`

Sets the arguments for the given problem. Each of the arguments within  $\langle args \rangle$  should be grouped, e.g. `\setprobargs{label}{5}{3}`. Inner braces are still required if there is only one argument, e.g. `\setprobargs{label}{{25}}`

```
\newcommand*\setprobargs[3][default]{%
\expandafter\gdef\csname prob@args@#1@#2\endcsname{#3}%
}
```

`\@setdefaultprobargs`

Like `\setprobargs` but only sets the arguments if `\usedefaultprobargstrue`. The database is given by `\thisprob@currentdb` (the current database) and the problem label is given by `\thisprob@currentlabel`. This command should only be used within `defproblem`.

```
\newcommand*\@setdefaultprobargs[1]{%
\ifusedefaultprobargs
\setprobargs[\thisprob@currentdb]{\thisprob@currentlabel}{#1}%
\fi
}
```

`\long@collect@body`

Need long versions of `\collect@body`. These macros are adapted from the macros defined by `amsmath`.

```
\long\def\long@collect@body#1{%
\@envbody{\@xp#1\@xp{\the\@envbody}}%
\edef\process@envbody{\the\@envbody\nx\end{\@currenvir}}%
\@envbody\@emptytoks \def\begin@stack{b}%
\begin@group
\@xp\let\csname\@currenvir\endcsname\long@collect@body
\edef\process@envbody{\@xp\nx\csname\@currenvir\endcsname}%
\process@envbody
}
```

`\long@addto@envbody`

```
\long\def\long@addto@envbody#1{%
  \toks@{#1}%
  \edef\@psn@tmp{\the\@envbody\the\toks@}%
  \global\@envbody\@xp{\@psn@tmp}%
}
```

`\long@collect@@body`

```
\long\def\long@collect@@body#1\end#2{%
  \protected@edef\begin@stack{%
    \long@push@begins#1\begin\end \@xp\@gobble\begin@stack
  }%
  \ifx\@empty\begin@stack
    \endgroup
    \@checkend{#2}%
    \long@addto@envbody{#1}%
  \else
    \long@addto@envbody{#1\end{#2}}%
  \fi
  \process@envbody
}
```

`\long@push@begins`

```
\long\def\long@push@begins#1\begin#2{%
  \ifx\end#2\else b\@xp\long@push@begins\fi
}
```

`defproblem`

```
\begin{defproblem}[\langle n \rangle][\langle default args \rangle]{\langle label \rangle}
```

Define a new problem identified by  $\langle label \rangle$  with  $\langle n \rangle$  arguments to add to the current data base. Note that since the contents of the environment are passed to a command, the contents can't contain any verbatim text.

```
\newenvironment{defproblem}[1][0]{%
  \edef\@prob@currentargN{\number0#1}%
  \@defproblem@beginenv
}{%}

\newcommand*{\@defproblem@beginenv}[2][]{%
  \def\@prob@currentdefaultargs{#1}%
  \def\@prob@currentlabel{#2}%
  \long@collect@body\prob@do@defproblem
}
```

`\prob@do@newproblem`

```
\prob@do@newproblem{\langle definition \rangle}
```

Defines a new problem given by  $\langle definition \rangle$ , where the number of arguments is given by  $\backslash@prob@currentargN$ , the label is given by  $\backslash@prob@currentlabel$  and the data base is given by  $\backslashprob@currentdb$ .

```
\newcommand{\prob@do@newproblem}[1]{%
```

```

\prob@newproblem\@prob@currentargN\prob@currentdb
\@prob@currentlabel{#1}{\@prob@currentdefaultargs}}

\prob@do@defproblem The default action of the defproblem environment is to define a new problem.
\let\prob@do@defproblem=\prob@do@newproblem

onlysolution Define an environment that only displays its contents if the solutions should be
displayed.
\newenvironment{onlysolution}{%
\long@collect@body\do@onlysolution
}{}

\do@onlysolution
\newcommand{\do@onlysolution}[1]{%
\ifshowanswers
#1%
\fi
Add to the list of solutions
\@ifundefined{@prob@currentlabel}
{}%
{%
\expandafter
\psn@add@unique@label
\csname prob@db@\prob@currentdb @solutions\endcsname{%
\@prob@currentlabel
}%
}%
}

\psn@add@unique@label \psn@add@unique@label{\list cs}{\label}}

```

Globally adds label to list if not already in list.

```

\newcommand*\psn@add@unique@label[2]{%
\ifx#1\@empty
\xdef#1{#2}%
\else
\edef\@tmp@label{#2}%
\expandafter\DTLifinlist\expandafter{\@tmp@label}{#1}%
{}% ignore
{\xdef#1{#1,\@tmp@label}}%
\fi
}
% \end{macrocode}
%\end{macro}
%
%\begin{macro}{\DTLifinlist}
% This is defined in \sty{datatool}, but there's no sense loading
% the entire package just for this, so define if not already
% defined.
%\changes{2011/08/22}{3.01}{new}
% \begin{macrocode}

```

```

\providecommand{\DTLifinlist}[4]{%
  \def\@dtl@doifinlist##1,#1,##2\end@dtl@doifinlist{%
    \def\@before{##1}%
    \def\@after{##2}%
  }%
  \expandafter\@dtl@doifinlist\expandafter,#2,#1,\@nil
  \end@dtl@doifinlist
  \ifx\@after\@nnil
% not found
  #4%
  \else
% found
  #3%
  \fi
}

```

`onlyproblem` Define an environment that only displays its contents if the solutions should not be displayed.

```

\newenvironment{onlyproblem}{%
  \long@collect@body\do@onlyproblem
}{}

```

`\do@onlyproblem`

```

\newcommand{\do@onlyproblem}[1]{%
  \ifshowanswers
  \else
  #1%
  \fi}

```

## 11.5 Using Problems

`\useproblem` `\useproblem[<db name>]{<label>}{<arg1>}...{<argN>}`

Use problem identified by *<label>* in data base *<db name>* where *<arg1>*...*<argN>* are the arguments to pass to the problem, if the problem was defined to take arguments.

```

\newcommand{\useproblem}[2][default]{%
  \def\@prob@currentlabel{#2}%
  \def\@prob@currentdb{#1}%
  \prob@showdraftlabel{#1}{#2}%
  \let\@useprob@next=\relax
  \@ifundefined{prob@data@#1@#2}{%
  {%
    \PackageError{probsoln}%
      {Problem ‘#2’ is not defined in data set ‘#1’}{-}%
  }%
  {%
    \def\@useprob@next{\csname prob@data@#1@#2\endcsname}%
  }%
  \@useprob@next
}

```

## 11.6 Loading Problems From Another File

`\loadallproblems` `\loadallproblems[ $\langle db name \rangle$ ]{ $\langle filename \rangle$ }`

Loads all the problems defined in  $\langle filename \rangle$  and adds them to data base  $\langle db name \rangle$ . (`\par` is temporarily disabled to allow for blank lines between problems.)

```
\newcommand*\loadallproblems[2][default]{%
\bgroup
\let\par\relax
\edef\prob@currentdb{#1}%
\input{#2}%
\egroup
}
```

`\prob@do@selectedproblem` Only define problem if the label is listed in `\prob@selectedlabels`. (The current label is given by `\@prob@currentlabel`.)

```
\newcommand{\prob@do@selectedproblem}[1]{%
\expandafter\DTLifinlist\expandafter{\@prob@currentlabel}{\prob@selectedlabels}%
{%
\prob@do@newproblem{#1}%
}%
{}%
}
```

`\loadselectedproblems` `\loadselectedproblems[ $\langle db name \rangle$ ]{ $\langle list \rangle$ }{ $\langle filename \rangle$ }`

Loads only those problems whose labels are listed in  $\langle list \rangle$ .

```
\newcommand{\loadselectedproblems}[3][default]{%
\bgroup
\let\par\relax
\edef\prob@currentdb{#1}%
\edef\prob@selectedlabels{#2}%
\let\prob@do@defproblem=\prob@do@selectedproblem
\input{#3}%
\egroup
}
```

`\prob@do@exceptedproblem` Only define problem if the label isn't listed in `\prob@selectedlabels`. (The current label is given by `\@prob@currentlabel`.)

```
\newcommand{\prob@do@exceptedproblem}[1]{%
\expandafter\DTLifinlist\expandafter{\@prob@currentlabel}{\prob@selectedlabels}%
{}%
{%
\prob@do@newproblem{#1}%
}%
}
```

`\loadexceptproblems` `\loadexceptproblems[ $\langle db name \rangle$ ]{ $\langle list \rangle$ }{ $\langle filename \rangle$ }`

Loads only those problems whose labels are not listed in  $\langle list \rangle$ .

```
\newcommand{\loadexceptproblems}[3] [default] {%
\bggroup
\let\par\relax
\edef\prob@currentdb{#1}%
\edef\prob@selectedlabels{#2}%
\let\prob@do@defproblem=\prob@do@exceptedproblem
\input{#3}%
\egroup
}
```

$\backslash$ prob@add@currentlabel Adds the current label to  $\backslash$ prob@selectedlabels (ignores argument.)

```
\newcommand{\prob@add@currentlabel}[1] {%
\ifx\prob@selectedlabels@empty
\edef\prob@selectedlabels{\@prob@currentlabel}%
\else
\edef\prob@selectedlabels{\prob@selectedlabels, \@prob@currentlabel}%
\fi
}
```

$\backslash$ iffirstpass Determines if this is the first pass of the filename when loading a data base.

```
\newif\iffirstpass
\firstpasstrue
```

$\backslash$ loadrandomproblems  $\backslash$ loadrandomproblems [ $\langle db\ name \rangle$ ] { $\langle n \rangle$ } { $\langle filename \rangle$ }

Loads  $\langle n \rangle$  randomly selected problems from  $\langle filename \rangle$ .

```
\newcommand{\loadrandomproblems}[3] [default] {%
\@loadrandomproblems{#1}{#2}{#3}{}%
}
```

$\backslash$ loadrandomexcept  $\backslash$ loadrandomexcept [ $\langle db\ name \rangle$ ] { $\langle n \rangle$ } { $\langle filename \rangle$ } { $\langle exception\ list \rangle$ }

Loads  $\langle n \rangle$  randomly selected problems from  $\langle filename \rangle$ , excluding labels listed in  $\langle exception\ list \rangle$ .

```
\newcommand{\loadrandomexcept}[4] [default] {%
\@loadrandomproblems{#1}{#2}{#3}{#4}%
}
```

$\backslash$ @loadrandomproblems Internal workings of  $\backslash$ loadrandomproblems and  $\backslash$ loadrandomexcept. Needs to input  $\langle filename \rangle$  twice: the first time just gathers all the labels, the second time only loads the selected problems.

```
\newcommand{\@loadrandomproblems}[4] {%
\bggroup
\let\par\relax
\def\prob@db@reserved{}%
\def\prob@currentdb{reserved}%
\edef\prob@selectedlabels{}%
```

Collect labels:

```
\let\prob@do@defproblem=\prob@add@currentlabel
\firstpasstrue
\input{#3}%
```

Shuffle labels.

```
\@probselN=0\relax
\@for\@thislabel:=\prob@selectedlabels\do{%
  \expandafter\DTLifinlist\expandafter
  {\@thislabel}{#4}{}%
  {%
    \advance\@probselN by 1\relax
    \expandafter
    \edef\csname @prob@tmp@\romannumeral\@probselN\endcsname{%
      \@thislabel}%
    }%
  }%
\shuffle{\@prob@tmp@}{\@probselN}%
\ifnum\@probselN<#2\relax

  \PackageWarning{probsoln}{You have requested
    \number#2\space\space problem(s) but ‘#3’ only contains
    \number\@probselN\space problems. All problems will be selected}%
\else
  \@probselN=#2\relax
\fi
```

Store only the first  $\langle n \rangle$  of the shuffled labels.

```
\@probN=0\relax
\def\prob@selectedlabels{%
\whiledo{\@probN<\@probselN}{%
  \advance\@probN by 1\relax
  \ifx\prob@selectedlabels\@empty
    \edef\prob@selectedlabels{%
      \csname @prob@tmp@\romannumeral\@probN\endcsname}%
  \else
    \edef\prob@selectedlabels{%
      \prob@selectedlabels,%
      \csname @prob@tmp@\romannumeral\@probN\endcsname}%
  \fi
}%
```

Only load selected labels.

```
\let\prob@do@defproblem=\prob@do@selectedproblem
\firstpassfalse
\input{#3}%
```

Move them from the reserved data base into the required data base in the order specified by `\prob@selectedlabels`

```
\@ifundefined{prob@db@#1}{\prob@newdb{#1}}{%
\@for\@thislabel:=\prob@selectedlabels\do{%
  \@moveproblem{\@thislabel}{reserved}{#1}%
}%
```

`\prob@selectedlabels` had to be globally defined. It's no longer required to undefine it.



```

\let\prob@selectedlabels=\undefined
\egroup
}

```

## 11.7 Iterating Through a Data Base

`\foreachproblem` `\foreachproblem[<db name>]{<body>}`

Does *<body>* for each problem defined in the data base *<db name>*. Within *<body>*, the command `\thisproblem` can be used to do the current problem and the command `\thisproblemlabel` can be used to access the current label.

```

\newcommand{\foreachproblem}[2][default]{%
\@ifundefined{prob@db@#1}{%
\PackageError{probsoln}{Data base ‘#1’ is not defined}{}%
}{%
\expandafter\let\expandafter\@tmp\csname prob@db@#1\endcsname
\@for\thisproblemlabel:=\@tmp\do{%
\expandafter\ifnum
\csname prob@argN@#1@\thisproblemlabel\endcsname>0\relax
\@ifundefined{prob@args@#1@\thisproblemlabel}{%
\expandafter\@prob@getargs
\csname prob@argN@#1@\thisproblemlabel\endcsname
{#1}\thisproblemlabel}}{%
\expandafter\let\expandafter\thisproblemargs
\csname prob@args@#1@\thisproblemlabel\endcsname
\else
\let\thisproblemargs\@empty
\fi
\expandafter\toks@\expandafter{\thisproblemargs}%
\edef\thisproblem{\noexpand\useproblem[#1]{\thisproblemlabel}%
\the\toks@}%
#2%
}%
}%
}

```

`\foreachsolution` `\foreachsolution[<db name>]{<body>}`

Does *<body>* for each problem defined in the data base *<db name>* that has a solution contained within `onlysolution`. Within *<body>*, the command `\thisproblem` can be used to do the current problem and the command `\thisproblemlabel` can be used to access the current label.

```

\newcommand{\foreachsolution}[2][default]{%
\@ifundefined{prob@db@#1}{%
\PackageError{probsoln}{Data base ‘#1’ is not defined}{}%
}{%
\expandafter\let\expandafter\@tmp\csname prob@db@#1@solutions\endcsname
\@for\thisproblemlabel:=\@tmp\do{%
\ifx\thisproblemlabel\@empty
\else

```

```

\expandafter\ifnum
  \csname prob@argN@#1@\thisproblemlabel\endcsname>0\relax
  \@ifundefined{prob@args@#1@\thisproblemlabel}{%
    \expandafter\@prob@getargs
    \csname prob@argN@#1@\thisproblemlabel\endcsname
    {#1}\thisproblemlabel}}}%
  \expandafter\let\expandafter\thisproblemargs
  \csname prob@args@#1@\thisproblemlabel\endcsname
\else
  \let\thisproblemargs\@empty
\fi
\expandafter\toks@\expandafter{\thisproblemargs}%
\edef\thisproblem{\noexpand\useproblem[#1]{\thisproblemlabel}%
  \the\toks@}%
#2%
\fi
}%
}%
}

```

`\foreachdataset` `\foreachdataset{<cmd>}{<body>}`

Iterates through all defined data sets. Assigns `<cmd>` to the name of the data base.

```

\newcommand{\foreachdataset}[2]{%
  \@for#1:=\prob@databases\do{#2}}

```

## 11.8 Random Numbers

First define some registers for later use.

```

\newcount\@probn \newcount\@probselN \newcount\@rndselctr
\newcount\@rndcur
\newcount\@ps@randtmp
\@rndcur=1\relax

```

`\PSNrandseed` Set the random generator seed

```

\newcommand*\PSNrandseed}[1]{%
  \ifnum#1=0\relax
    \PackageWarning{probsoln}{Can't have 0 as random seed, changing to 1}%
    \global\@rndcur=1\relax
  \else
    \global\@rndcur=#1\relax
  \fi
  \PackageInfo{probsoln}{Random Seed = \number\@rndcur}%
}

```

`\PSNgetrandseed`

```

\newcommand*\PSNgetrandseed}[1]{#1=\@rndcur\relax}

```

`\PSNrand` Generate a random integer.

```

\newcommand*\PSNrand{%
  \@ps@randtmp=\@rndcur
}

```

```

\multiply\@ps@randtmp by 16811\relax
\r@ndcur=\@ps@randtmp
\global\divide\r@ndcur by 39989\relax
\global\multiply\r@ndcur by 39989\relax
\advance\@ps@randtmp by -\r@ndcur
\global\r@ndcur = \@ps@randtmp
\ifnum\r@ndcur=0\relax
  \global\r@ndcur=1\relax
\fi
}

```

`\PSNold@rand` Random generator used in v3.0 and earlier

```

\newcommand*\PSNold@rand}{%
  \@ps@randtmp=\r@ndcur
  \multiply\@ps@randtmp by 16807\relax
  \r@ndcur=\@ps@randtmp
  \global\divide\r@ndcur by 12001\relax
  \global\multiply\r@ndcur by 12001\relax
  \advance\@ps@randtmp by -\r@ndcur
  \global\r@ndcur = \@ps@randtmp
  \ifnum\r@ndcur=0\relax
    \global\r@ndcur=1\relax
  \fi
}

```

`\PSNuseoldrandom` Use the old random number generator

```

\newcommand*\PSNuseoldrandom}{%
  \let\PSNrand\PSNold@rand
}

```

`\PSNrandom` `\PSNrandom{<count>}{<n>}`

stores a random number from 1 to  $\langle n \rangle$  in the TeX count register  $\langle count \rangle$

```

\newcommand*\PSNrandom}[2]{%

```

generate new random number.

```

  \PSNrand
  #1=\r@ndcur
  \@ps@randtmp=\r@ndcur

```

now set  $\langle count \rangle$  to  $(\langle count \rangle \bmod \langle n \rangle) + 1$

```

  \divide\@ps@randtmp by #2\relax
  \multiply\@ps@randtmp by #2\relax
  \advance#1 by -\@ps@randtmp
  \advance#1 by 1\relax
}

```

`\random` `\random{<counter>}{<a>}{<b>}`

Generate a random number in the range  $[a, b]$ , and store this number in the L<sup>A</sup>T<sub>E</sub>X counter  $\langle counter \rangle$ .

```

\newcommand{\random}[3]{%
\ifnum#2=1\relax
  \PSNrandom{\value{#1}}{#3}%
\else
  \@rndselctr=#3%
  \advance\@rndselctr by -#2\relax
  \advance\@rndselctr by 1\relax
  \PSNrandom{\value{#1}}{\@rndselctr}%
  \addtocounter{#1}{#2}%
  \addtocounter{#1}{-1}%
\fi
}

```

`\shuffle` Shuffle contents of pseudo-array. For example, suppose you have the following definitions: `\def\fooi{A}`, `\def\fooi{B}` and `\def\fooi{C}`, then `\shuffle{foo}{3}` will shuffle the definitions, so you may end up with, e.g. `\def\fooi{C}`, `\def\fooi{A}`, `\def\fooi{B}`, or some other variation.

```

\newcount\@shfctr \newcount\@shfA \newcount\@shfB
\newcommand{\shuffle}[2]{%
  \@shfctr=1\relax
  \whiledo{\@shfctr < 101}{%
    {%
      \PSNrandom{\@shfA}{#2}\PSNrandom{\@shfB}{#2}%
      \ifnum\@shfA=\@shfB
        \else
          \edef\@tmpA{\csname#1\romannumeral\@shfA\endcsname}%
          \let\@tmpA=\@tmpA
          \edef\@tmpB{\csname#1\romannumeral\@shfB\endcsname}%
          \let\@tmpB=\@tmpB
          \expandafter\xdef\csname#1\romannumeral\@shfA\endcsname{\@tmpB}%
          \expandafter\xdef\csname#1\romannumeral\@shfB\endcsname{\@tmpA}%
        \fi
        \advance\@shfctr by 1\relax
      }%
    }
}

```

`\doforrandN` `\doforrandN{<n>}{<cmd>}{<list>}{<text>}`.

A bit like `\@for` but only for a random subset of the given list. For example, the following will load one problem each from two out of the three listed files.

```

\doforrandN{2}{\tmp}{file1,file2,file3}{%
\loadrandomproblems{1}{\tmp}}

\newcount\@ps@forrand
\newcommand{\doforrandN}[4]{%
  {%
    \@ps@forrand=0\relax
    \@for#2:=#3\do{%
      \advance\@ps@forrand by 1\relax
      \expandafter
        \edef\csname @doforrandN\romannumeral\@ps@forrand\endcsname{#2}%
    }%
  }%
}

```

```

\ifnum\@ps@forrand<#1\relax
  \PackageError{probsoln}{Can't randomly select \number#1 item(s)}{You
  have requested \number#1 item(s), but there
  are only \number\@ps@forrand item(s) in the list}%
\else
  \shuffle{@doforrandN@}{\@ps@forrand}%
  \ifnum#1>0\relax
    \@ps@forrand=0\relax
  \loop
    \advance\@ps@forrand by 1\relax
    \edef#2{\csname @doforrandN@\romannumeral\@ps@forrand\endcsname}%
    #4%
  \ifnum\@ps@forrand<#1\relax
    \repeat
  \fi
\fi
}%
}

```

## 11.9 Compatibility With Older Versions

These commands ensure that this version is compatible with versions prior to v3.0

```

\newproblem Defines a new problem.
  \newcommand*{\newproblem}{\@ifstar\@snewproblem\@newproblem}

\@snewproblem Store first optional argument
  \newcommand{\@snewproblem}[1][0]{%
  \def\@newprob@argN{#1}%
  \@s@newproblem
  }%

\@s@newproblem Define a new problem without a solution
  \newcommand{\@s@newproblem}[3][]{%
  \begin{defproblem}[\@newprob@argN][#1]{#2}%
  #3%
  \end{defproblem}%
  }

\@newproblem Store first optional argument
  \newcommand{\@newproblem}[1][0]{%
  \def\@newprob@argN{#1}%
  \@ns@newproblem
  }

\@ns@newproblem Define problem with a solution:
  \newcommand{\@ns@newproblem}[4][]{%
  \begin{defproblem}[\@newprob@argN][#1]{#2}%
  #3%
  \begin{onlysolution}%
  \begin{solution}%
  #4%
  \end{solution}%
  \end{defproblem}%
  }

```

```

        \end{onlysolution}%
    \end{defproblem}%
}

\selectallproblems
    \newcommand*\selectallproblems[1]{\loadallproblems[#1]{#1}}%
    \foreachproblem[#1]{\PSNitem\thisproblem\endPSNitem}}

\selectrandomly
    \newcommand*\selectrandomly[2]{%
    {\loadrandomproblems[#1]{#2}{#1}}%
    \foreachproblem[#1]{\PSNitem\thisproblem\endPSNitem}%
    }

PSNitem
    \newenvironment{PSNitem}{\item}{}

```

## 11.10 Formatting Commands

These commands are provided to format parts of the problems/solutions.

```

solution
    \@ifundefined{solution}{%
    \newenvironment{solution}{\par\noindent\textbf{\solutionname:}\space
    \ignorespaces}{}%
    }{}

```

```

\solutionname
    \newcommand*\solutionname{Solution}

```

Define an in-line enumeration which uses the enumeration environment's counters.

```

textenum
    \newenvironment{textenum}{%
    \ifnum\@enumdepth>\thr@@
    \toodeep
    \else
    \advance\@enumdepth\@ne\relax
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
    \let\@item\@textitem
    \def\@itemlabel{%
    \refstepcounter{\@enumctr}%
    \csname label\@enumctr\endcsname
    }%
    \setcounter{\@enumctr}{0}%
    \fi
    \ignorespaces
    }%
    {%
    \global\advance\@enumdepth\m@ne\relax
    }

\@textitem In-line enumeration item
    \def\@textitem[#1]{#1\space\ignorespaces}

```

`\correctitemformat` Indicates how to format the item label for `\correctitem` when the solutions are shown. The argument is the label. This defaults to placing the argument in a box.  
`\newcommand*{\correctitemformat}[1]{\fbox{#1}}`

`\incorrectitemformat` Indicates how to format the item label for `\incorrectitem` when the solutions are shown. The argument is the label. This defaults to just the argument shifted by `\fboxsep + \fboxrule` to ensure it aligns with the default `\correctitemformat`.  
`\newcommand*{\incorrectitemformat}[1]{%`  
`\hspace{\fboxsep}\hspace{\fboxrule}#1}`

`\correctitem` This can be used instead of `\item`. If the solutions are not shown, it behaves like `\item`, otherwise, it's like `\item`, but the label is formatted according to `\correctitemformat`.  
`\newcommand*{\correctitem}{\@inmatherr\correctitem`  
`\@ifnextchar[\@correctitem{\@noitemargtrue\@correctitem[\@itemlabel]}}`  
`\def\@correctitem[#1]{%`  
`\ifshowanswers`  
`\@item[\correctitemformat{#1}]%`  
`\else`  
`\@item[#1]%`  
`\fi}`

`\incorrectitem` This can be used instead of `\item`. If the solutions are not shown, it behaves like `\item`, otherwise, it's like `\item`, but the label is formatted according to `\incorrectitemformat`.  
`\newcommand*{\incorrectitem}{\@inmatherr\incorrectitem`  
`\@ifnextchar[\@incorrectitem{\@noitemargtrue\@incorrectitem[\@itemlabel]}}`  
`\def\@incorrectitem[#1]{%`  
`\ifshowanswers`  
`\@item[\incorrectitemformat{#1}]%`  
`\else`  
`\@item[#1]%`  
`\fi}`

# Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the definition; numbers in *roman* refer to the pages where the entry is used.

Symbols	F	
<code>\loadrandomproblems</code> <a href="#">23</a>	<code>\foreachdataset</code> <a href="#">.. 9, 26</a>	<code>onlysolution</code> (environment) <a href="#">... 3, 3,</a> <a href="#">5, 9, 10, 14, 20, 25</a>
<code>\moveproblem</code> <a href="#">..... 16</a>	<code>\foreachproblem</code> <a href="#">6, 9, 25</a>	P
<code>\newproblem</code> <a href="#">..... 29</a>	<code>\foreachsolution</code> <a href="#">3, 9, 25</a>	package options:
<code>\ns@newproblm</code> <a href="#">... 29</a>	fp package <a href="#">..... 6</a>	<code>answers</code> <a href="#">..... 2, 9, 14</a>
<code>\prob@do@getargs</code> <a href="#">. 18</a>	H	<code>draft</code> <a href="#">..... 2</a>
<code>\prob@getargs</code> <a href="#">... 17</a>	<code>\hideanswers</code> <a href="#">2, 5, 10, 14</a>	<code>final</code> <a href="#">..... 2</a>
<code>\prob@gobblethree</code> <a href="#">. 17</a>	I	<code>noanswers</code> <a href="#">..... 2, 14</a>
<code>\s@newproblem</code> <a href="#">... 29</a>	<code>\iffirstpass</code> <a href="#">..... 23</a>	<code>nousedefaultargs</code> <a href="#">... 2</a>
<code>\setdefaultprobargs</code> <a href="#">18</a>	<code>\ifshowanswers</code> <a href="#">... 3, 14</a>	<code>usedefaultargs</code> <a href="#">. 2, 6, 9</a>
<code>\snewproblem</code> <a href="#">..... 29</a>	ifthen package <a href="#">..... 2</a>	<code>usedefaultargs</code> <a href="#">. 14</a>
<code>\textitem</code> <a href="#">..... 30</a>	<code>\ifusedefaultprobargs</code> <a href="#">..... 14</a>	pgfmath package <a href="#">..... 6</a>
A	<code>\include</code> <a href="#">..... 5</a>	<code>\prob@add@currentlabel</code> <a href="#">..... 23</a>
<code>alltt</code> package <a href="#">..... 3, 7</a>	<code>\incorrectitem</code> <a href="#">... 4, 31</a>	<code>\prob@currentdb</code> <a href="#">... 15</a>
<code>amsmath</code> package <a href="#">... 18</a>	<code>\incorrectitemformat</code> <a href="#">..... 4, 31</a>	<code>\prob@databases</code> <a href="#">... 15</a>
B	<code>\input</code> <a href="#">..... 5, 8</a>	<code>\prob@do@defproblem</code> <a href="#">20</a>
<code>babel</code> package <a href="#">..... 5</a>	<code>\item</code> <a href="#">..... 4</a>	<code>\prob@do@exceptedproblem</code> <a href="#">..... 22</a>
C	L	<code>\prob@do@newproblem</code> <a href="#">19</a>
<code>\correctitem</code> <a href="#">..... 4, 31</a>	<code>\loadallproblems</code> <a href="#">. 8, 22</a>	<code>\prob@do@selectedproblem</code> <a href="#">..... 22</a>
<code>\correctitemformat</code> <a href="#">4, 31</a>	<code>\loadexceptproblems</code> <a href="#">..... 8, 22</a>	<code>\prob@newdb</code> <a href="#">..... 15</a>
D	<code>\loadrandomexcept</code> <a href="#">8, 23</a>	<code>\prob@newproblem</code> <a href="#">.. 17</a>
<code>defproblem</code> (environment) <a href="#">..... 3,</a> <a href="#">5, 5, 7, 9, 18-20</a>	<code>\loadrandomproblems</code> <a href="#">..... 8, 11, 23</a>	<code>\prob@showdraftlabel</code> <a href="#">14</a>
<code>\do@onlyproblem</code> <a href="#">... 21</a>	<code>\loadselectedproblems</code> <a href="#">..... 8, 22</a>	<code>probsoln</code> package <a href="#">... 12</a>
<code>\do@onlysolution</code> <a href="#">.. 20</a>	<code>\long@addto@envbody</code> <a href="#">19</a>	<code>\psn@add@unique@label</code> <a href="#">..... 20</a>
<code>\doforrandN</code> <a href="#">..... 12, 28</a>	<code>\long@collect@body</code> <a href="#">19</a>	<code>\PSN@old@rand</code> <a href="#">..... 27</a>
<code>\draftproblemlabel</code> <a href="#">. 14</a>	<code>\long@collect@body</code> <a href="#">. 18</a>	<code>\PSNgetrandseed</code> <a href="#">. 11, 26</a>
E	<code>\long@push@begins</code> <a href="#">. 19</a>	<code>PSNitem</code> (environment) <a href="#">30</a>
<code>enumerate</code> (environment) <a href="#">..... 4, 10</a>	M	<code>\PSNrand</code> <a href="#">..... 26</a>
environments:	<code>\moveproblem</code> <a href="#">..... 15</a>	<code>\PSNrandom</code> <a href="#">..... 11, 27</a>
<code>defproblem</code> <a href="#">.....</a> <a href="#">3, 5, 5, 7, 9, 18-20</a>	N	<code>\PSNrandseed</code> <a href="#">..... 11, 26</a>
<code>enumerate</code> <a href="#">..... 4, 10</a>	<code>\newproblem</code> <a href="#">6, 7, 9, 12, 29</a>	<code>\PSNuseoldrandom</code> <a href="#">..</a> <a href="#">..... 8, 11, 27</a>
<code>onlyproblem</code> <a href="#">.....</a> <a href="#">. 3, 3, 5, 10, 14, 21</a>	<code>\newproblem*</code> <a href="#">... 7, 9, 12</a>	R
<code>onlysolution</code> <a href="#">3, 3,</a> <a href="#">5, 9, 10, 14, 20, 25</a>	ngerman package <a href="#">..... 5</a>	<code>\random</code> <a href="#">..... 12, 27</a>
<code>PSNitem</code> <a href="#">..... 30</a>	O	S
<code>solution</code> <a href="#">... 3, 3, 30</a>	<code>onlyproblem</code> (environment) <a href="#">.....</a> <a href="#">. 3, 3, 5, 10, 14, 21</a>	<code>\selectallproblems</code> <a href="#">. 13, 30</a>
<code>textenum</code> <a href="#">..... 3, 30</a>		<code>\selectrandomly</code> <a href="#">... 12, 13, 30</a>
		<code>\setprobargs</code> <a href="#">..... 18</a>



<code>\showanswers</code> .....	<code>\solutionname</code> .... <a href="#">3</a> , <a href="#">30</a>	<b>U</b>
..... <a href="#">2</a> , <a href="#">5</a> , <a href="#">9</a> , <a href="#">10</a> , <a href="#">14</a>		
<code>showanswers</code> boolean	<b>T</b>	<code>usedefaultargs</code> (op-
variable .. <a href="#">2</a> , <a href="#">3</a> , <a href="#">5</a>	<code>textenum</code> (envir-	tion) ..... <a href="#">14</a>
	ment) ..... <a href="#">3</a> , <a href="#">30</a>	<code>\usedefaultprobargstrue</code>
<code>\shuffle</code> .....	<code>\theenumi</code> .....	..... <a href="#">18</a>
	<code>\thisproblem</code> .... <a href="#">2</a> , <a href="#">6</a> , <a href="#">9</a>	
<code>solution</code> (envir-	<code>\thisproblemlabel</code> .. <a href="#">9</a>	<code>\useproblem</code> .... <a href="#">7</a> , <a href="#">9</a> , <a href="#">21</a>
ment) ... <a href="#">3</a> , <a href="#">3</a> , <a href="#">30</a>		