The esami package for examinations*

Grazia Messineo and Salvatore Vassallo

Abstract

The package esami is a small collection of macros to prepare written examinations and tests for students at universities or secondary schools. It generates output in which questions and answers are scrambled. Exercises depend on random parameters, the values of which are assigned during the compilation and on which you can do many arithmetic operations.

1 Introduction

Among the main topics on tex.stackexchange.com are questions about writing exams with many different versions, scrambling questions or lists, hiding answers in tests, etc. We wrote this package in order to solve some of these problems for written maths examinations at the Faculty of Economy of Catholic University in Milan.

We began to develop the package in 2008 and we tried to extend some useful properties of the LaTeX packages exerquiz by D. P. Story [4] and probsoln by N. Talbot [5]. In particular we liked the idea of creating a database of exercises from which we could select one or more. (The reader should know that we began work with the 2008 versions of these packages and never updated them.) Moreover we needed to be able to use random parameters in exercises and to generate many (from 12 to 100 or more) different, but similar, versions of the same assignment. Over the years, the development of our package has had several points in common with the development of Dr. Story's package exerquiz and others; we found similar solutions, but with different code.

The package was developed having mathematics in mind, but can be used in different fields.

2 Our goal

Our goal was the creation of a simple database of exercises: many files, including many variants on several "basic" ones; the exercises can depend on random parameters and they can be multiple choice questions (MCQ) or problems, i.e. exercises with solutions, simple or divided in multiple parts. When the examination is generated, the process is:

- selection of the files with the exercises,
- random choice of the version of the exercise in the files,
- shuffle of the exercises chosen,
- shuffle of the answers in the MCQ,
- assignment of values to random parameters.

Moreover, we need a file with the solutions of the exercises and, for any version, a string (or more) with the correct choice for the MCQ. Some of these facilities were already contained in the package exerquiz (and some have been introduced by Dr. Story in later versions of the package). The choice of the variant of an exercise was solved by changing a command of the package probsoln. We wanted the package to be straightforward to use for users with very little knowledge of LaTeX.

3 The exercises typology

Our aim was to use two types of exercises: "tests", that is, sets of multiple choice questions, and "problems", that is, exercises with an articulate development, such as the study of a function or the discussion and solution of a linear system. But we also wanted to be able to use the same software in different situations, so we decided to implement other typologies of "exercises":

(a) questions with a short answer where the student has to write the answer in a provided space at the end of the exercise;
(b) "*fill-in*" questions; in which some blank space has to be filled (for example in theorems);
(c) questions with a "long" answer;
(d) tables to be completed;
(e) "*matching*": exercises where the student has to connect the elements of two lists (like nations and capitals).

Naturally, all these exercises can contain random parameters, can be shuffled and can have a different appearance.

4 The randomization problem

Instead of using only the package random for the random choices, we preferred to use a "mixed" method. The shuffling of the answers in the MCQ and choosing of values of random parameters are totally random. On the other hand, the choice of a variant of an exercise is random if the number of variants is greater than a predetermined value (currently set at 8); otherwise, a permutation among 24 possible permutations (6 if there are 3 variants) is chosen (deterministically, based on the version number of the exam). A similar procedure is also used for choosing the order of the exercises in the task.

The seed of the randomization process is based on a combination of the exam date and the version

* Work supported by Catholic University Research Project "Progetto M.In.E.R.Va: Creazione di esercizi di tipo interattivo con percorsi differenziati per il recupero delle carenze e la valorizzazione delle eccellenze in matematica".

number, such that different versions on the same date are essentially different and the "same" exams given on different dates are different. Moreover, when a value is given to a random parameter the seed is changed using the order number of the exercise, so that parameters defined in the same interval, but in different exercises, are not all equal.

5 The exam

In order to generate an assignment we need some files besides the `.sty` file:

(a) the files with the exercises chosen from the database;

(b) a "master" file in which the user has to write the date of the exam, the number of versions to generate, and the name of the files of the exercises; in this file it is also possible to change the appearance, for example the geometry of the page, or the size of the font (for this, some little LaTeX knowledge is needed);

(c) another "master" file for solutions similar to the previous one, that generates the solutions for any version of the exam and the string of the correct answers to the MCQ;

(d) a configuration file with commands for the footers, the headers, the geometry of the page, the instructions for the students, etc.

One more optional file can be used to check the database: for any file of exercises it prints all the variants both in numeric and in parametric format. This uses an option that modifies the way in which the mathematical expressions are elaborated by LaTeX.

So the task of the teachers is the creation of the database of exercises written using the instructions of the package.

We had many reasons for this type of structure:

(a) the process of production and checking of the exercises is completely independent from the generation of an assignment, even if the format of the files is the same: this allows the creation of a database of exercises that can be increased as the teachers have time;

(b) the appearance of the assignment can be modified without any change to the `.sty` file;

(c) users with little or no knowledge of LaTeX can generate the assignment if the database is sufficient;

(d) with some small modifications to the master file it's possible to obtain exams in one or more parts, with different kinds and numbers of exercises, etc.

6 The database of exercises

Every exercise with all its variants is written in a separate file; each variant is enclosed in the command `\newproblem`, a highly modified version of the (2008 version) command with the same name in the package `probsoln`. This command has just one argument: the text of the exercise itself.

6.1 Multiple choice questions

If the exercise is an MCQ the syntax is almost the same as that of `exerquiz`:

```
\item \PTs{⟨points⟩}
 ... Exercise text ...
\begin{answers}{⟨number-of-columns⟩}
    \bChoices[random]
\Ans0 incorrect answer \eAns
\Ans0 incorrect answer \eAns
\Ans1 correct answer \eAns
    \eFreeze
\Ans0 none of the preceding \eAns
    \eChoices
\end{answers}
```

where:

- The '`\item \PTs{`⟨*points*⟩`}`' introduces a question with a score of `\PTs` points[1] (it can also be a decimal number and the separator can be the comma, unlike in `exerquiz`);
- `\begin{answers}{`⟨*number-of-columns*⟩`}`
  `\bChoices[random]`
  `...`
  `\eChoices`
  `\end{answers}`
  typesets the answers in ⟨*number-of-columns*⟩; if there is the option `random`, the answers are randomly shuffled;
- `\Ans0` indicates an incorrect answer;
- `\Ans1` indicates a correct answer;
- `\eFreeze`: after this command the answers are not randomized, and they appear at the end of the list.

Unfortunately, the method we used to obtain the string with the correct answers to a set of MCQ has for now excluded the possibility of using questions with more than one correct answer or with answers having different scores.

6.2 Open exercises

If the exercise is an open exercise (i.e. an exercise with a complete solution), it is embedded in the

[1] Since the package is written for Italian users, the default label is "punto" or "punti" (Italian words for "point" and "points"): this can be changed using the macro `\PTsHook`.

environment `problem` or `problem*` (if it has one or more parts). The syntax is:

```
\begin{problem}[⟨score⟩]
 ... Text of the exercise ...
\begin{solution}[⟨space-for-sol⟩]
 ... solution ...
\end{solution}
\end{problem}
```

where ⟨*space-for-sol*⟩ is the height of the (optional) blank space left for the solution and ⟨*score*⟩ is the score of the exercise. If it is an exercise with multiple parts:

```
\begin{problem*}[⟨total-score⟩]
... text ...
\begin{parts}
\item \PTs{⟨partial-score⟩}
... text ...
   \begin{solution}[⟨space-for-sol⟩]
    ... text of solution ...
    \end{solution}
\item \PTs{⟨partial-score⟩}
...
\end{parts}  \end{problem*}
```

where `\PTs{`⟨*partial-score*⟩`}` is the score of each part. The package `exerquiz` has the facility of automatically calculating the total score of an exercise. In our package, due to the shuffling of the exercises, this is not always possible.

6.3 Other types of exercises

The other types of exercises we defined are:

fill-in For creating exercises in which some text is left blank and must be filled in by the student, or exercises with an open short answer. The syntax is:
`\fillin[`⟨*type*⟩`]{`⟨*width-of-blank*⟩`}{`⟨*answer*⟩`}`
The two mandatory parameters are the width of the blank space, expressed as a length, and the correct answer — text or number — that the student has to write: it will be printed in the solutions only. The optional parameter ⟨*type*⟩ defines the way the blank space is denoted: `u` (*underlined*), the default, produces an underlined space; `b` (*boxed*) produces a little box; `e` (*empty*) produces an empty space. In the blank space it's not possible to use the commands for the simplifications (see Section 9).

Example 1

```
The capital of Italy is
\fillin[u]{5cm}{Rome},
the capital of France is
\fillin[b]{4cm}{Paris}
```

The capital of Italy is _____, the capital of France is [ ]

matching This is based on an idea from the package `examdesign` [1]. It is used to create exercises in which the student has to match items in two lists. The pairs are defined with '`{`⟨*item1*⟩`}{`⟨*item2*⟩`}`', repeated for each pair of items to match. The two lists are shuffled and then printed with the command `\matching`.

Example 2

```
\pair{Italy}{Rome}
\pair{Germany}{Berlin}
\pair{Greece}{Athens}
\matching
```

_____	Greece	(A) Berlin
_____	Italy	(B) Athens
_____	Germany	(C) Rome

The solution shows the correct matching.

tabella This is used to create exercises with many short open answers in a column. The syntax is (the `\cr` at the end of the line is necessary):

```
\begin{tabella}[⟨num-visible-cols⟩]
   {⟨visible-cols-align⟩}
   {⟨hidden-col-align⟩}
... & ... \cr
\end{tabella}
```

The optional parameter (default 2) is the number of columns of the table visible in the text of the exercise. The last column is invisible in the text and visible in the solutions. The second parameter gives the alignment of the visible columns (the same for all the columns) and the third the alignment of the hidden column.

Example 3

```
\begin{center}
\renewcommand\arraystretch{3}
\begin{tabella}[1]{l}{l}
 \hline
   The domain of the function is:
   & $D=(-\infty;2]$ \cr
 \hline
   The range of $f(x)$ is:
   & $f(D)=(-\infty,0]$ \cr
 \hline
\end{tabella}
\end{center}
```

we obtain (the second column is visible only in the solutions):

The domain of the function is:	$D = (-\infty; 2]$
The range of $f(x)$ is:	$f(D) = (-\infty, 0]$

The following environments don't define exercises, but help to format or check the exercises.

`problema` **and** `problema*` These environments are like `problem` and `problem*`, but if the package option `solutionsonly` is specified, only the solution of the exercise is printed and not the text.

`risposta` This environment generates a ruled or boxed space in which the student has to write the answer to an exercise ("risposta" is the Italian word for "answer"). The syntax of the command is:

```
\begin{risposta}{⟨type⟩}{⟨vertical-space⟩}
 ...
\end{risposta}
```

The ⟨*type*⟩ parameter defines if the blank space has to be boxed (option `b`, the default) or ruled (option `l`). The parameter ⟨*vertical-space*⟩ defines the height of the space for the answer: it is a length if it is boxed or the number of rules if it is ruled.

`workarea` This environment defines a blank space on the paper sheet where the student can write. In this space it's possible to put some text, a graphic, coordinate axis, etc. The syntax is:

```
\begin{solution}{⟨height⟩}
\end{solution}

\begin{workarea}[⟨width⟩]{⟨height⟩}
\end{workarea}
```

The height of the `solution` and `workarea` environments should be equal; if the `workarea` height is larger, the text of the `workarea` will be misaligned in the space of the solution, overlapping with the exercise. The width of the `workarea` is optional and by default is equal to the textwidth.

7 The master files

7.1 The file `master` and `master-sol`

The only difference between these two files is that the second one shows the solutions. They contain all the instructions to generate the exam. In both files it's necessary to write:

- the (same) date in the (same) format, namely ⟨*day*⟩/⟨*month*⟩/⟨*year*⟩ (the day and month can be in any format, the year should be written with four digits: 3/12/2012, 03/7/2013),
- the name of the exercises (command `\esercizi`),
- the number of versions (command `\numcompiti`).

The exam can be in multiple parts and in any part it's possible to use one or more of the environments defined above and one or more commands for the choice of the exercises. In the file there is also the definition of the random seed (command `\seme`). It is also possible to use the classical sectioning commands.

In these files the MCQ are embedded in the environment `test` with the optional parameter ⟨*score*⟩. In this environment there are one or more sets of MCQ, each introduced by `\begin{questions}`.

The other kinds of exercises can be contained or not in the `\begin{questions}` environment, except that `problem` and similar cannot be there. However, although fill-in exercises with more than one blank to fill and matching exercises can be used in a `test` environment, the string of correct answers at the end of the file is no longer useful because the numbering of questions is wrong. If you are not interested in the final string of correct answers, you can use them without any problem. (See also the `fillb` package option described later.)

7.2 The file `totale-versioni`

The file `totale-versioni` (i.e. all the versions) is used to generate all the versions of an exercise that are in a file of the database and it's desired to check them. In this case the master must have the option `prova` (see Section 8); the compilation gives a numeric version and, with the option `param` the parametric version (with the random parameters not evaluated) of the exercises.

The `totale-version` file itself has just one command, `\def\esercizio{`⟨*file*⟩`}`, where ⟨*file*⟩ is the name of the file of exercises. When compiling the parametric version the name of the parameters and their range of variation will be printed. The file works similarly to the command `\selectallproblems` of the package `probsoln`.

8 Package options

The package `esami` has many options:

- `allowrandomize` and `norandomize`: with the first the answers in MCQ are shuffled (default), with the second they are printed in the order they are written;
- `shuffle`, `shufflerandom` and `noshuffle`: the first (default) shuffles the exercises (randomly if there are more than eight, in a deterministic way if there are 8 or less), with the second the exercises are always shuffled randomly (by uncommenting some lines in the file `esami.sty` it's possible to make the choice be random for more than $n < 8$ exercises and deterministic otherwise), with the third the exercises are not shuffled at all;

- `xxxx`: reads the file '`esami-xxxx.cfg`' that contains some commands and configurations, such as the name of the course, instructions for the students, etc. The names of some configuration files are given in the file `esami.sty`, but it's possible to read another configuration file without modifying anything: it's sufficient to put a unknown option like `zzz` and create the file `esami-zzz.cfg`;
- `pointsonright`: a boolean option that generates a little box on the right of the page with the score of the exercise
- `nosolutions`: with this option the exam is generated without solutions (default);
- `solutions`: generates the file of solutions;
- `solutionsonly`: generates a file with solutions only if the environment `problema` is used;
- `prova`: as mentioned above, when compiling the file `totale-versioni` with this option, a PDF file is generated with all the variants of an exercise; the correct answers of all MCQ and the solutions of the exercises are automatically shown;
- `param`: with this option, used only in conjunction with the option `prova`, the versions of the exercise are printed in parametric form; it also shows the range of variation of the parameters;
- `correzione`: can be used only with the option `prova`, to print only the text of all the exercises, without solutions;
- `fillb`: this option is necessary to have the correct answers in the string of solutions if there are exercises of `fillin` type;
- `twocolumns`: with this option, the MCQ are printed in two columns;
- `sansserif`: a sans serif font is used.
- `autopston` and `autopstoff`: both these options load the package `auto-pst-pdf`, in the second case with the option `off`; in this way it's possible to compile the file directly with PDFLaTeX even if the exercises contain graphics in `pstricks` — the graphics package we use. With the first option, the images are generated and included in the document, while the second doesn't generate the images but includes them if they exist.

9 Package commands

9.1 Commands working with parameters

As we said above, one of the goals of the package is to use random parameters in exercises. We defined only integer parameters but it is possible to define also rational or (pseudo)real parameters, as D.P. Story does in the package `rangen` [3]. Since we use the package `fp` [2] to do calculations, almost all the commands operating on parameters are prefixed by FP. The command to define a parameter is `\FPsetpar[`⟨*seed*⟩`]{`⟨*param-name*⟩`}{`⟨*inf*⟩`}{`⟨*sup*⟩`}[`⟨*excl-values*⟩`]`.

- the name of the random parameter will be the control sequence \⟨*param-name*⟩;
- the parameter's range will be between ⟨*inf*⟩ and ⟨*sup*⟩ (inclusive);
- the optional ⟨*seed*⟩ is used to have a different seed for the generation of the random number, with a default value given by `\seme` (the Italian word for seed) defined in the preamble;
- one or more values can be excluded from the choice with ⟨*excl-values*⟩. If there is more than one excluded value, the whole list is enclosed in braces.

The lower and the upper bounds (⟨*inf*⟩ and ⟨*sup*⟩, with ⟨*inf*⟩ < ⟨*sup*⟩) and the excluded values can be random parameters defined earlier. In order to satisfy the conditions the generation of the random number may be repeated many times; the maximum number of repetitions is given by the command `\maxLoopLimit`, by default 10 (this can be redefined in the preamble of the document).

Example 4

```
\FPsetpar{a}{2}{10}[3]
\FPsetpar{b}{4}{12}[{\a,6}]
```

generates two random numbers `\a` (with range between 2 and 10, but not 3) and `\b` (with range between 4 and 12, excluding both the value assigned to `\a` and 6).

We defined some commands in addition to those in the `fp` package to do operations on parameters.

The command `\FPsv[`⟨*decimal*⟩`]{`⟨*operation*⟩`}` is used to evaluate ⟨*operation*⟩ (on numbers or parameters) obtaining either the numeric value with ⟨*decimal*⟩ decimal places (by default 0 decimal places) or, with the package option `param`, the typesetting of the operation.

Example 5 `\FPsv{2*k+1}`, with (say) $k = 2$, gives either 5 or, with the option `param`, $2 * k + 1$; `\FPsv[2]{(2*k+1)/2}` gives 2.50 or $(2 * k + 1)/2$.

The syntax of the arithmetic operations is the same as in the package `fp`. When used with `param`, it's easier to read if the operations are given in parentheses.

The command `\FPval{`⟨*name*⟩`}{`⟨*operation*⟩`}` assigns to \⟨*name*⟩ the rounded result of ⟨*operation*⟩. (This is a modified form of the command `\FPeval` from `fp`.)

Example 6
```
\FPsetpar{k}{1}{3}
\FPval{a}{2*k+1}
\FPsetpar{b}{2}{20}[\a]
```
generates a random parameter `\b` which assumes a value between 2 and 20, but different from `\a`, where `\a` is given by `2*k+1`. In the parametric version it will appear like this:
The parameter b varies from 2 to 20. $b \neq (2 * k + 1)$.

We also defined some commands to simplify fractions, that can also be used for correct formatting of the text.

The command `\sempli{`⟨*num*⟩`}{`⟨*den*⟩`}` simplifies a fraction where ⟨*num*⟩ and ⟨*den*⟩ can contain parameters or operations on them.

Example 7 If $k = 1$, `\sempli{2*k}{3*k+1}` gives $\frac{1}{2}$ or, with `param` specified, $\frac{2*k}{3*k+1}$.

The command `\semplix{`⟨*num*⟩`}{`⟨*den*⟩`}` simplifies a fraction where ⟨*num*⟩ and ⟨*den*⟩ can contain parameters, but where the result 1 does not appear and the result -1 is shown as just a minus sign "$-$" (for example to be used before an x). This command can also be used to format coefficients of a variable, setting the denominator equal to 1.

Example 8 If $k = 2$, `\FPsv{k-1}x` gives $1x$ while `\semplix{k-1}{1}x` gives just x.

The command `\esempli{`⟨*num*⟩`}{`⟨*den*⟩`}` simplifies a fraction such that the result 1 does not appear, and the result -1 has to appear explicitly (as in exponents). The command can be used with a denominator of 1 to correctly format the exponents.

Example 9 If $k = 2$, `x^{\FPsv{k-1}}` gives x^1 while `x^{\esempli{k-1}{1}}` gives just x.

The command `\sempliz{`⟨*num*⟩`}{`⟨*den*⟩`}` simplifies fractions that can assume the value 0; with the other commands, the result 0 gives an error and stops the compilation.

The command `\simpsqrt{`⟨*ind*⟩`}{`⟨*rad*⟩`}` allows extracting factors from radicals; however, it is not possible to do other operations with these factors. The first mandatory parameter ⟨*ind*⟩ is the index of the radical and can be parametric; the second one, ⟨*rad*⟩ is the radicand and can be also a parameter or an operation.

Example 10 If $a = 2$ and $b = 1$, `\simpsqrt{2}{a^2+4*b}` gives $2\sqrt{2}$.

9.2 Commands for exercises and lists

The commands to manage both exercises and lists are in the same category since they work in the same way: given a list of tokens, they shuffle the objects and pick some elements.

The main command to manage exercises is:
`\esercizi{`⟨*file1*⟩`, `⟨*file2*⟩`, ..., `⟨*fileN*⟩`}`
This chooses a random exercise for each given file, shuffles them and sends the result to the output.

The command `\estrai[`⟨*m*⟩`]{`⟨*list*⟩`}{`⟨*name*⟩`}`, with ⟨*list*⟩ being a comma separated list of n objects, picks $n - m$ elements from ⟨*list*⟩; the selected elements will be called `\`⟨*name*⟩`i`, `\`⟨*name*⟩`ii`, and so on; they can be used, for example, within the command `\esercizi`.

Example 11
```
\estrai[2]{sets,log,exp}{arg}
```
This chooses two elements of the given list, setting `\argi` and `\argii` to the values: by writing `\esercizi{\argi,\argii}` we obtain two random exercises, one from each of the two randomly-chosen topics (sets, logarithms, and exponentials).

The command `\estraialfa{`⟨*n*⟩`}{`⟨*list*⟩`}{`⟨*name*⟩`}` similarly picks ⟨*n*⟩ random objects from ⟨*list*⟩, but preserving the order. As before, the elements will be called `\`⟨*name*⟩`i`, `\`⟨*name*⟩`ii`, etc.

Example 12
```
\estraialfa[2]{a,b,c,d}{alpha}
```
This chooses two elements from the given set of four letters, while preserving alphabetical order. The chosen elements are stored in `\alphai` and `\alphaii`.

Finally, the command `\estraies[`⟨*m*⟩`]{`⟨*list*⟩`}` also works similarly to the command `\estrai`, but only on an exercise's list; the chosen elements go to the output instead of being stored.

With these commands we can have many different possibilities of random choice.

It's possible to use the commands `\esercizi` and/or `\estraies` many times. This is useful if one would like to have exercises from two or more different subsets (for example, 5 exercises about limits chosen from 7 available, and 3 about derivatives chosen from 5) or, more simply, if one likes to have some exercises in two columns and others in one column.

10 Open issues

In the package there remain open issues. The code can be improved and made more efficient, in particular in the management of the lists — for example using `etoolbox` — and the solution we found to obtain the string of the solutions of the MCQ, using

\label and \ref and the aux file, doesn't allow for questions with more than one correct answer.

Other improvements can be made from an aesthetic point of view: in particular, we decided to put each MCQ in a minipage to avoid misunderstandings by the students if a question was split across pages. As a result, the output is sometimes very ugly.

11 Examples of the working files

To conclude, here is a set of small complete files. First, an exercise file test1.tex, with one MCQ:

```
\newproblem{ \FPsetpar{a}{2}{5}
\item \PTs{1} exercise 1a
\begin{answers}{1}\bChoices[random]
\Ans1 answer 1 correct\eAns
\Ans0 answer 2 wrong\eAns
\Ans0 answer 3 wrong\eAns
\eChoices\end{answers}}
```

Next, the file totale-versioni used to print all versions of an exercise file (in this case, with MCQ):

```
\documentclass[english]{article}
\usepackage[mg,prova,param]{esami}
    %% make parametric version;
    %% for the numeric version, omit 'param'
\date{30/4/2008}  %% for the seed
\begin{document}
\FPeval\seme{209} %% or some other number
\randomi=\seme
\def\esercizio{test1}  %% exercise file
\begin{center} %% the title
 \makeatletter \ifes@param
  {\textbf{\esercizio -p}}
  \else {\textbf{\esercizio}}\fi
 \vspace{5mm}\end{center}
\begin{shortquiz}  % for MCQ
 \begin{questions} % for MCQ
  \selectallproblems{\esercizio}
 \end{questions}
\end{shortquiz}
\end{document}
```

Finally, an example of master file for generation of the exam (or solutions). The commented-out commands are for solutions.

```
\documentclass[english]{article}
\usepackage[test,shuffle,nosolutions]{esami}
%\usepackage[test,shuffle,solutions]{esami}
```

with these options a configuration file esami-test.cfg is read, the exercises are shuffled, the figures are not generated and the text is printed in one column.

```
\def\numcompiti{10} %% How many versions?
\date{17/02/2012}
\begin{document}
\date{\Data}
\pagestyle{esame} %% defined in cfg file
%\immediate\openout\sols=\thenomefile.sol.tex
        %% for solutions
\whiledo{\thevers<\numcompiti}{\stepcounter{vers}
    %% the routine to generate the versions
\FPeval\seme{round((\thenomefile+\thevers):0)}
    %% the random seed can be anything;
    %% \thenomefile ''is'' the date
\randomi=\seme
%\immediate\write\sols{\string\begin{minipage}
%{.3\textwidth}Solution of Version \thevers}
    %% for solutions
\testa %% the header defined in cfg file
\section*{Part one}
%\immediate\write\sols{\string\subsection*{Part
% one} \par\string\begin{enumerate}}
    %% for solutions
\begin{test}[6] %% MCQ for a total of 6 points
 \begin{questions}
  \esercizi{test1}
 \end{questions}
\end{test}
%\immediate\write\sols{\par\end{enumerate}
\string\end{minipage}\par}%% for  solutions
\section*{Part two}
...
}
%\immediate\closeout\sols %% for solutions
%\stringasol
\end{document}
```

References

[1] Jason Alexander. The package examdesign. mirror.ctan.org/macros/latex/contrib/examdesign, 2006.

[2] Michael Mehlich. The package fp. mirror.ctan.org/macros/latex/contrib/fp, 1999.

[3] D.P. Story. The package rangen. www.math.uakron.edu/~dpstory/rangen.html, 2009.

[4] D.P. Story. Exerquiz & AcroTEX. www.acrotex.net, 2012.

[5] Nicola L.C. Talbot. The package probsoln. mirror.ctan.org/macros/latex/contrib/probsoln, 2011.

◇ Grazia Messineo
Università Cattolica Milano
Largo Gemelli, 1
Milan, I-20123; and
ITC "G. Falcone"
Viale Italia, 22
Corsico, I-20094 Italy
grazia dot messineo (at) unicatt dot it

◇ Salvatore Vassallo
Università Cattolica Milano
Largo Gemelli, 1
Milan, I-20123
salvatore dot vassallo (at) unicatt dot it