Package esami*

Grazia Messineo <grazia.messineo@unicatt.it>
Salvatore Vassallo <salvatore.vassallo@unicatt.it>

July 21, 2023

Contents

*This document refers to the package esami versione 2.8, with date 2023/07/21.

1 Introduction and motivations

Package `esami` (i.e. exams in Italian) was created to generate texts and solutions for the exams of Mathematics in the Faculty of Economics of Catholic University of Milan and satisfies the need of generating many versions (usually from 12 to 50) of the same exam, containing similar, but different, exercises, of printing solutions and, if Multiple Choice Questions (MCQ) are inserted, the string of correct answers.

The package allows to write various type of exercises (multiple choiche questions with answers varying in random order, with closed or open answer, *matching*, problems, and so on). Questions of each group are randomly scrambled across the exam and variants of each exercise are chosen randomly from a file which contains them all and, as far as we know, it is the only package which allows to do this just using LaTeX ad not auxiliary programs.

The development of the package has begun in 2008 ([5], [4]) with the aim of unify and extend some properties of the packages `exerquiz` e `eqexam` by D.P. Story [7] and `probsoln` by N.Talbot [8].

2 Required packages

Package `esami` depends on some other non standard packages[1]:

1. `fp`: for mathematical operations;
2. `currfile`: to obtain the file and folder names;
3. `xargs`: to have more than one optional parameter;
4. `environ`: to transform commands in environments;

Moreover following packages are optional, but recommended:

1. `ifluatex` and `ifxetex` to compile with XeLaTeX and LuaLaTeX (here below we will refer only to the compilation with pdfLaTeX)[2]. New vers. 2.0
2. `lmodern` and `amsfonts`: useful for a good pdf output;
3. `numprint`: to have a better output and different formats for numbers, depending on language;
4. `eurosym`: (optional) to use the euro symbol in Financial Mathematics exercises;
5. `icomma`: from the package `ws` for a correct formatting of commas. This package can be substituted, for the Italian language, by the macro `\IntelligentComma` of the package `babel`: this is the default option in the italian localization file `es-italian.lng` and can be modified by commenting the macro `\IntelligentComma` (or by using the macro `\NoIntelligentComma`); New vers. 1.1
6. `pstricks, pstricks-add` and `auto-pst-pdf` or `tikz` or another package for graphics. New vers. 2.0

All these packages are available both in TeXLive and in MiKTeX.

[1] The complete list of the packages used in a normal compilation of an exam is: etex, graphicx, enumerate, fp, currfile, array, environ, ifthen, xargs, xkeyval, multicol, amssymb, amsfonts, xcolor, babel, etoolbox, icomma, numprint, amsmath, fontenc, inputenc, lmodern. Obviously, every loaded package can have other dependencies.

[2] At the moment the use of package with XeLaTeX and LuaLaTeXhas not completely been tested and it is possibile that some type of exercises do not work as expected.

3 Installation

The package contains the file `esami.sty`, some configuration files `*.cfg`, the localization files, 3 templates for the compilation and the documentation files. File `sty` and files `cfg` and `lng` must be copied in a folder where LaTeX can find them (i.e. `texmf-local/tex/latex/esami`). An update of the database must be run.

4 How to write an exam

In this section we will see how to build an exam using the "esami" package.

4.1 Preamble

```
\documentclass[language]{article}
\usepackage[<options>]{esami}
```

Options are discussed in section 5.

```
\def\numcompiti{10}
\date{2018/04/21}%%% THE DATE IN THE FORMAT YYYY/MM/DD
\def\Data{\longdate}%%% or \shortdate: the date in the heading
```

Command `\date` is the standard LaTeX command, but the date must be written in the form YYYY/MM/DD. Command `\numcompiti` states the number of versions to generate. Command `\Date` now prints the date in the form DD MMMM YYYY (`\longdate`) or DD/MM/YYYY

4.2 The document

```
\begin{document}
\pagestyle{esame}
\setcounter{vers}{m}
```

Command `\pagestyle` is defined in a configuration file (see section 8.2) and defines headers and footers. The commented command `\setcounter`{vers}{m} is used, jointly to the command `\numcompiti` with inside a number n (greater than m), to generate the versions from $m+1$ to n.

```
\whiledo{\thevers<\numcompiti}{\stepvers
```

the routine for the compilation of variants begins here and it will end at the end of document.

```
\FPeval\seme{round((\thenomefile+\thevers):0)}
\randomi=\seme
```

generates the seed for the randomization process and assigns it also to `\randomi`.

New vers. 2.2

N.B. If the seed is a number smaller than 1, it is initialized using date and hour, so two compilations will give different outputs. It is now possible to define in a different way the seed for permutations (`\seme`) and the one for the choice of the exercises (`\semeex`): the need arises from the request to be able to choose for each exam the same exercises for all the students (`\semeex` depends only on the date), but also to be able to permute them in each version.

```
\permuta
\testa
\istruzionii
```

`\testa`, `\istruzionii` are defined in the configuration file and contain the instructions for the heading and the instructions for the exam, while the macro `\permuta` allows the randomization of exercises order.

```
\begin{test}[points]
\begin{questions}
\esercizi{test1,test2,test3,test4}
\begin{esdb}{dbone}
 \selectrandomlyn{2}{dbtest1}
\selectrandomlyn{4}{dbtest2}
\end{esdb}
\end{questions}
\end{test}
```

Here we define an environment for a test with (usually) MCQ. The command `\esercizi` chooses randomly the exercises from the files written in it and scrambles them, the environment esdb defines a virtual database of exercises in which they are put by the command `\selectrandomlyn`.

```
  \esercizi{problema1, problema2}
```

chooses between two exercises with open answer.

```
\closevers
}
%\def\stringasol{}
\end{document}
```

It ends the routine for the compilation. The macro `\def\stringasol{}` is used to avoid the
New vers. 2.0 generation of the string of correct answers in the solution file, if you don't want it.

The compilation produces a pdf file with all the desired variants. The compilation of the solutions file produces also another file, namely `filename-sol.loe` (loe = list of exercises) which contains the labels assigned to the chosen exercises, to identify them: each chosen exercise is identified by a string such as *e:number of version -file:name of file-q:number of*
New vers. 2.1 *question in roman digits*.

At the end of the solutions file there is the string of the correct answers for multiple choice questions: as a default behaviour this string is, for each version, included in a minipage environment. If this choice is not suitable for our purposes (i. e. if we have 40 multiple choice questions and the answers cannot be contained in a page) it is possibile to modify it adapting the macros `\stepverssols` and `\closeverssols`: in the example cfg file there are (commented) the definitions to have a continuous string of correct answers.

During the compilation, we have introduced a control that checks if the number of points marked for a test or an exercises divided in parts is equal to the sum of the partial points. This possibility can be excluded by commenting in the configuration file the definition of
New vers. 2.1 the command `\checkpoints`.

5 Package options

The package can be loaded with the usual command `\usepackage`[⟨*options*⟩]{esami}. The possibile options are:

- *allowrandomize* and *norandomize*: with the first one the answers in MCQ are shuffled (default), with the second one they are printed in the same order in which they are written;

- *shuffle*, *shufflerandom* and *noshuffle*: the first one (the default) shuffles the exercises (randomly if they are more than eight, in a deterministic way if they are 8 or less), with the second one the exercises are always shuffled randomly (uncommenting some lines in the file `esami.sty` it is possible to modify it so that the choice is random for more than $n < 8$ exercises and deterministic otherwise), with the third the exercises are not shuffled;
- *xyz*: it reads the file (`esami-xyz.cfg`) that contains some commands and configurations, such as the name of the course, the instructions for the students, etc. The name of some configuration files is written in the file `esami.sty`, but it is possible to read a configuration file without modifying it: it is sufficient to put an unknown option like *zzz* and write the file `esami-zzz.cfg`;
- *pointsonright*: it is 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*: it generates the solutions file;

New vers. 2.0
- *noquizsolutions*: it generates the test without solutions in MCQ;

New vers. 2.0
- *quizsolutions*: it generates the test with the solutions in MCQ;
- *prova*: it is the Italian word for trial. Compiling the file `totale-versioni` with this option, a PDF file is generated with all the variants of an exercise; automatically the correct answers of MCQ and the solutions of the exercises are shown;
- *param*: with this option, used only in conjunction with the option *prova*, the versions of the exercise are printed in parametric form, moreover it shows the range of variation of the parameters;
- *correzione*: it is the Italian word for correction. It can be used only with the option *prova*. It prints only the text of all the exercises, without solution;

New vers. 2.0
- *fillb*: deprecated, it remains for compatibility with previous versions. It is used with "fillin" exercises, but it is recommended to use the macro `\fillinproblem` (see 6.4)
- *twocolumns*: with this option the multiple choice questions are printed in two columns;
- *sansserif*: a *sans serif* font is used.
- *autopston* e *autopstoff*: both these options load the packages `pstricks`, `pstricks-add` and `auto-pst-pdf` but the second one with the option *off*; in this way it is possible to compile the file directly with pdfLaTeX also if the exercises contain graphics in `pstricks`, the graphics package we use. With the first one the images are generated and included in the document, while the second one does not generate the images but includes them if they exist.[3].
- *language*: it reads the localisation file es-language.lng (see 8.1). In this file there are the translations of the labels (exercise, solution, ...), of the error messages, of the headings contained in the file (`esami-xyz.cfg`). By default Italian language is chosen. Since these language options have the same name of the languages in `babel` it's possible to put the option in `\documentclass`.

[3]The package `auto-pst-pdf` requires pdfLaTeX to be called with the `shell-escape` (in TeXLive) or `write18` (in MiKTeX) option

6 Exercises

6.1 Variants

`\newproblem` Each exercise (with all its variants) must be written in a separate file. Each variant is enclosed in the command `\newproblem{ ...}` (which has no optional arguments).

Example 1.

```
%% This is file1.tex with 2 variants of an exercise
\newproblem{Text of first variant}
\newproblem{Text of second variant}
```

Command `\newproblem` is a highly modified version of the similar command of the package `probsoln`.

6.2 The environment test

`test` (*env.*) The environment test allows to write exercises with MCQ or with open short answer, in which each question is randomly chosen from one of the files containing all the exercises and their variants. The environment is created by the command

```
\begin{test}[<points>]
  ...
\end{test}
```

`points` The optional parameter `[`⟨*points*⟩`]` represents the total number of points given to the whole group of questions.

`questions` (*env.*) The questions of each group, in the master file, are enclosed in the environment *questions* which is a modified version of the environment with the same name in the package `exerquiz`:

```
\begin{questions}
    ....
\end{questions}
```

It is possibile to insert in a test environment more than one questions environment. The exercises of each group will be independently scrambled.

Example 2.

```
\begin{test}
    \begin{questions}
        ....
    \end{questions}
....
    \begin{questions}
        ....
    \end{questions}
\end{test}
```

`\esercizi` Inside this environment, questions are "loaded" in two ways: with the environment esdb which will be described later, or with the command `\esercizi` (exercises in Italian):

```
\esercizi{name of source file 1, name of source file 2,...}
```

The mandatory parameter is the name of each exercise you want to insert into the test, separated by a comma[4].

[4] We will describe other similar commands for loading the exercises in subsection 7.2.

It is possibile both to insert all the exercises in a unique command and to use more than one command `\esercizi` or `\estraies` or `\randestraies` (see 7.2 and 7.2). This is useful for example if you want exercises from two different subsets (5 exercises on limits chosen among 7 and 3 exercises on derivatives chosen among 5) or if you want to have some exercises on two columns and other on one column, for a better layout of the page. In this case you must precede the part on two columns with `\begin{multicols}{2}` and end it with `\end{multicols}`. You **must not** use option *twocolumns*.

Example 3.

```
\begin{test}[punti]
\begin{multicols}{2}
\esercizi{es1,es2,es3}
\end{multicols}
\esercizi{es4,es5,es6}
\end{test}
```

In this way, exercises `es1,es2,es3` are typeset on two columns, while exercises `es4,es5,es6` are typeset on one column.

esdb (*env.*)
selectrandomlyn

The environment `esdb{`⟨*nome db*⟩`}` defines a virtual database of exercises named *nome db* (the name of the database **must** be different from the name of every other file of exercises, of other databases and from the name given to the master file, because it is the name of a file written during the compilation). Within this environment, one or more exercises can be chosen with the commands `\selectrandomlyn{`⟨*numero esercizi*⟩`}{`⟨*nome file*⟩`}`, where *nome file* is the name of the file from which exercises are extracted and *numero esercizi* is the number of exercises to be extracted (if the number of exercises is "all" or is greater than
New vers. 2.0 the number of variants in the file, all the exercises are chosen).

The environment generates a file `nome db.tex` which contains the references to all the exercises chosen. When the environment is closed, exercises are automatically written with the command `\esercizidb{`⟨*nome db*⟩`}` (not to be written). If more than one environment esdb is used, each one must be given a different name.

Also this environment is a modified version of a similar environment in `probsoln`.

In the file of the exercises, in `\newproblem` you must write:

```
 \item \PTs{points}
...text...
\begin{answers}{number of columns}
    \bChoices[random]
\Ans0 wrong answer \eAns
\Ans0 wrong answer \eAns
\Ans1 correct answer \eAns
        \eFreeze
\Ans0 none of the other answers is correct \eAns
    \eChoices
\end{answers}
```

in which:

- `\item \PTs{points}`

  \PTs introduces a question with points written in the command `\PTs` (it can be a decimal number and the separator can be the comma).

- `\begin{answers}{number of columns}`
  `\bChoices[random]`

```
    ...
    \eChoices
\end{answers}
```

`answers` (*env.*) introduces answers split on the specified number of columns. Answers will be randomly shuffled only if the option *random* is specified.

`\Ans0` `\Ans1` `\bChoices` `\eChoices` `\eFreeze`

- `\Ans0` introduces a wrong answer
- `\Ans1` introduces the correct answer
- `\eFreeze` introduces (if desireded) one or more answers which will not be randomly shuffled.

`solution` (*env.*) New vers. 2.0 The environment solution can be inserted *after* the environment answers and shows, using the option *quizsolutions*, the solutions of the exercise.

Example 4.

```
\newproblem{
\item \PTs{1} The solutions of the equation
\[x^2-5x+6=0\]
are
\begin{answers}{2}
    \bChoices[random]
\Ans0 $x=3$ e $x=-2$ \eAns
\Ans0 $x=-3$ e $x=-2$  \eAns
\Ans1 $x=3$ e $x=2$  \eAns
        \eFreeze
\Ans0 none of the other answers is correct \eAns
    \eChoices
\end{answers}
}

\newproblem{
\item \PTs{1} The solutions of the equation
\[x^2+5x+6=0\]
are
\begin{answers}{2}
    \bChoices[random]
\Ans0 $x=3$ e $x=-2$ \eAns
\Ans1 $x=-3$ e $x=-2$  \eAns
\Ans0 $x=3$ e $x=2$  \eAns
        \eFreeze
\Ans0 none of the other answers is correct \eAns
    \eChoices
\end{answers}
}
```

Besides MCQ, you can use in the environment test other kinds of questions, that will be described in section 6.4. Unfortunately using `fill-in` type exercises, if you have more than one blank space to fill, the string of correct answers will be less useful because the numbering of the questions is wrong.

N.B. By default each question of the test is enclosed in minipage environment. In the example cfg file there are some commented lines which allow to have questions out of this environment.

6.3 Environments problem and problem*

problem (*env.*)
problem* (*env.*)

These environments are used to write problems with open answers. They are used in the file which contains the variants of the exercise, nested in the command `\newproblem`, with the following syntax:

```
\begin{problem}
  ...
\end{problem}
```

if you choose to write an open answer exercise with only one part; with the syntax

```
\begin{problem*}
  ...
\end{problem*}
```

if you choose to write an open answer exercise with more than one part. The code for these environments is taken, with various modifications, from the code of the environment exercise in the package `exerquiz`[5].

Exercises in one part It is introduced by the environment problem.

```
\begin{problem} [score]
........Text..........
\begin{solution}[space for the solution]
.... text of solution ......
\end{solution}
\end{problem}
```

solution (*env.*)

The argument `[`⟨*score*⟩`]` contains the points of the exercise, the argument `[`⟨*space for the solution*⟩`]` contains the dimension of the *possible* white space which must be left for the solution.

parts (*env.*)

Exercise in more than one part It is introduced by the environment problem*.

```
\begin{problem*} [score]
....text....
\begin{parts}
\item \PTs{partial score}
....text....
   \begin{solution}[space for the solution]
    .... text of solution ......
    \end{solution}
\item \PTs{partial score}
.....
\end{parts}  \end{problem*}
```

The argument `\PTs{partial score}` contains the points of each part.

problemmp (*env.*)
problemmp* (*env.*)
New vers. 2.1

Exercises in a minipage We have also created the environments problemmp and problemmp* which are identical to the environments problem and problem*, but enclose all the text of the problem in a minipage environment.

[5] The environment exercise still exists and can be used as a template to build other environments such as new exercises, examples, etc.

6.4 Other kinds of exercise

We have defined other new kinds of exercise:

`\fillin` **fillin:** it is used to create exercises in which some parts of the text is left blank and must be filled by the student. It can also be used to create exercises with an open short answer. This kind of exercise is similar to the one introduced by D. P. Story in `eqexam`.

These exercises are introduced with a slightly different syntax and they are defined in the text by the command `\newfillinproblem` or by `\newproblem\fillinproblem{text of the problem}`. The syntax is:

New vers. 2.0

`\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 only in the solutions. The optional parameter ⟨*type*⟩ defines the way in which 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.

If the answer contains mathematical expressions, it must be written between $ even if the command `\fillin` is already contained in a mathematical environment.

`domanda` (*env.*) **domanda:** (the word "domanda" means "question" in Italian) this environment must contain the text of the exercise (not the solution). It is used in problem and problem* and with the package option *solutionsonly* only the solution of the exercise is printed and not the text. In previous versions of this package (not released to CTAN) there was another environment in order to obtain this.

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

```
[\small]
\begin{risposta}{type}{vertical_space}
 ...
\end{risposta}
```

The parameter `type` 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.

`\matching` **matching:** it is based on an idea of the package `examdesign` [1]. It is used to create exercises in which the student has to match items in two lists.

`\pair` The pairs are defined by the command `\pair`:

```
\pair{item 1}{item 2}
```

repeated for each couple of items to match.

The two lists are shuffled and then printed with the command `\matching`.

Example 5.

```
\pair{United Kingdom}{London}
\pair{France}{Paris}
\pair{Italy}{Rome}
\pair{Germany}{Berlin}
\pair{Greece}{Athens}
\matching
```

_____	Greece	(A) Paris
_____	United Kingdom	(B) Berlin
_____	France	(C) Rome
_____	Italy	(D) London
_____	Germany	(E) Athens

The solutions show the correct matching.

`tabella` (*env.*) **tabella:** (the word "tabella" means "table" in Italian) it is used to create exercises with many short open answers in 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 first 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 one the alignment of the hidden column.

Example 6. With the code

```
\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 in the solutions only):

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

`workarea` (*env.*) **workarea:** this environment defines a blank space on the paper sheet where the student can write. The syntax is:

```
\begin{solution}{height}
\end{solution}
\begin{workarea}[width]{height}
\end{workarea}
```

The height of workarea and of solution should be equal; on the contrary, if the text of the workarea is greater than the height of the solution, it will be misaligned in the space of the solution, overlapping to the exercise. The width of the workarea is optional and by default is equal to the textwidth. Differently from the environment solution, in the blank space it's possible to put some text, a graphic, coordinate axis, etc.

6.5 Other commands

The package defines other two commands used in the file `totale-versioni` (see 8.4)

\selectallproblems
- `\selectallproblems{\esercizio}`: it is used in the file which shows all the variants of an exercise, with calculated or uncalculated parameters.

\esercizio
- `\esercizio`: it is used in the file which shows all the variants of an exercise and defines the name of the exercise on that you are working and which is the argument of the command `\selectallproblems`. It will be printed as the title of the generated file. The syntax is `\def\esercizio{name}`.

7 Commands to work with parameters and lists

7.1 Commands to work on (random) parameters

We have defined some commands to work with (random) parameters.

These commands are based on the package `fp` [3], which allows the execution of (complex) operations inside a LaTeX document.

\FPsetpar
The basic command to define a parameter is `\FPsetpar[⟨seed⟩]{⟨parameter-name⟩}{⟨first value⟩}{⟨last value⟩}[⟨excl-values⟩]`. The name of the random parameter will be`\param-name` and its range will be between `inf` and `sup` (included).

The parameter will be an integer number.[6] The optional parameter `[⟨seed⟩]` is used to have a different seed for the generation of the random number. The default value is `\seme` (the Italian word for seed): it is defined in the preamble and it is based on the date of the exam and on the number of version.

It's possible to exclude one or more values from the choice (parameter `[⟨excl-values⟩]`). If the excluded values are more than one, they are enclosed in braces. The lower and the upper bounds (`{⟨inf⟩}` and `{⟨sup⟩}`, with `{⟨inf⟩}` < `{⟨sup⟩}`) and the excluded values can be random parameters defined before. In order to satisfy the conditions, the generation of the random number is repeated many times: the maximum number of repetitions is given by the command `\maxLoopLimit`, by default 10 (it can be redefined in the preamble of the document).

Example 7.

```
\FPsetpar{a}{2}{10}
```

creates a parameter `\a` which can be a random value between 2 and 10, in this case, the seed is defined in the preamble of the document. The commands

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

create two parameters, `\a` (which can be a random value between 2 and 10, except 3) and `\b` (which can be a random value between 4 and 12, except 6 and the value assigned to `\a`).

N.B. It is better to define the parameters inside the environments problem, problem*, etc, to avoid spurious spaces between the name of the exercise and its body text. Anyway, the authors suggest to begin the text on a new line.

[6] Even if it is possible to define rational or (pseudo)real parameters, as in the package `rangen` [6], we preferred to limit the choice to integer numbers and to obtain the other cases with operations on parameters.

You can work on the parameters defined with this command as on numbers and we have defined commands which allow to work on them showing both the numerical result and the operations in a parametric form.

\FPsv

- `\FPsv[`⟨*decimal places*⟩`]{`⟨*operation*⟩`}`: it is used to do operations (on numbers or parameters) obtaining or the numeric value with ⟨*decimal*⟩ decimal places (by default 0 decimal places), eliminating the useless zeroes or, with the option *param* in the package, the typesetting of the operation (if you choose the option *param* in the package `esami`).

Example 8.

Instruction	Numerical result	Option `param`
`\FPsv{2*k+1}`	5	$2 * k + 1$
`\FPsv[2]{(2*k+1)/2}`	2.50	$(2 * k + 1)/2$

The fundamental characteristics of the command `\FPsv` are (in the examples, $k = 2$):

- the operations must be written explicitly

  Example 9.

  `$\FPsv{2*k}$` 4

- each operation is automatically enclosed in parenthesis. The parenthesis is not printed if the numerical value is shown, but it is printed when you use the option *param*

  Example 10.

Instruction	Numerical result	Option `param`
`$\FPsv{2*k+1}x$`	$5x$	$(2 * k + 1)x$

- you can use the following symbols for operations: `+,-,*,/,^`
- Attention must be paid to the fact that the package `fp` cannot handle powers with a negative basis.

\FPval

- `\FPval{`⟨*name*⟩`}[`⟨*decimal places*⟩`]{`⟨*parameter/operation on parameters*⟩`}`: it assigns to `{`⟨*nome*⟩`}` the rounded value of the operation (it is the command `\FPeval` from `fp`, modified) or it prints the operation if you choose the option *param* in the package `esami`, as with `\FPsv`. The result is given with the chosen number of decimal places, eliminating the useless zeroes.

New vers. 2.0

Example 11. Let $k = 2$. The code

```
\FPsetpar{k}{1}{3}
  \FPval{a}{2*k+1}
  \FPsetpar{b}{2}{20}[\a]
```

generates a parameter `\b` which assumes a random value between 2 and 20, with the exception (in this case) of the value 5. In the parametric version, a similar string appears (as far as the parameter `\b` is concerned)

The parameter b varies between 2 and 20. $b \neq (2 * k + 1)$. The seed is 209.

\sempli

- \sempli{⟨*num*⟩}{⟨*den*⟩}: it simplifies a fraction where ⟨*num*⟩ and ⟨*den*⟩ can contain parameters or operations on them. Inside the command \sempli you must not use \FPsv.

Example 12.
If $k = 1$:

`$\sempli{2*k}{3*k+1}$`	$\frac{1}{2}$
`$\frac{\FPsv{2*k}}{\FPsv{3*k+1}}$`	$\frac{2}{4}$

In the log file it is possible to see if and in which exercises the result of this operation is 1 or -1 (see below).

\semplix

- \semplix{⟨*num*⟩}{⟨*den*⟩}: it simplifies a fraction where {⟨*num*⟩} and {⟨*den*⟩} can contain parameters, but where the result 1 does not appear and the result -1 is shown like a minus sign "$-$" (for example it can be used before a x).

  It works exactly as the command \sempli.

  The command can also be used, setting the denominator equal to 1, to format coefficients of a variable, so that the value 1 does not appear and the value -1 appears as $-$.

Example 13.
If the parameter k is equal to 2:

`$\FPsv{k-1}x$`	$1x$
`$\semplix{k-1}{1}x$`	x
`$\semplix{1-k}{1}x$`	$-x$
`$\semplix{2*k}{k+2}x$`	x

\esempli

- \esempli{⟨*num*⟩}{⟨*den*⟩}: it simplifies a fraction so that the result 1 cannot appear, but the result -1 has to appear explicitly (like in exponents). The command can be used also with denominator equal to 1 to format the exponents correctly.

  It works exactly as the command \sempli.

  The command can also be used, setting the denominator equal to 1, to format exponents, so that the value 1 does not appear and the value -1 appears as -1.

Example 14.
If the parameter k is equal to 2:

`$x^{\FPsv{k-1}}$`	x^{1}
`$x^{\esempli{k-1}{1}}$`	x
`$x^{\esempli{1-k}{1}}$`	x^{-1}

\sempliz

- \sempliz{⟨*num*⟩}{⟨*den*⟩}: simplifies fractions that can assume the value 0 (for example, in answers). While with the other commands the result 0 gives an **error** and stops the compilation, with this command the value 0 is written.

  It works exactly as the command \sempli.

\simpsqrt

- \simpsqrt{⟨*ind*⟩}{⟨*rad*⟩}: With this command it's possible to extract factors from radicals. With these factors is not possible to do other operations. 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 (also in this case \FPsv must not be used inside it).

Example 15.
If a is equal to 2 and b is equal to 1
`$\simpsqrt{2}{a^2+4*b}$` $2\sqrt{2}$

\RandS

- \RandS: it is used to give a random sign (+ o −) to a number. The command is similar to the one of the package rangen.

\FPsignpol

- \FPsignpol{⟨*expression*⟩}: gives the value of "expression" with its sign: it is useful, for example, if "expression" is the coefficient of a polynomial. This macro has been contributed by Hjalmar Basile.

New vers. 2.1

Example 16.
If a is 2 and b is -1
`$x^2\FPsignpol{a+4*b}{x}-2$` $x^2 - 2x - 2$
If a is 2 and b is 1
`$x^2\FPsignpol{a+4*b}{x}-2$` $x^2 + 6x - 2$

7.2 Commands to choose from a list

Besides the command \esercizi that we have already presented there are other commands to choose exercises, or, more in general, objects, from a list.

\estrai The first command allows to choose randomly $n - m$ elements of a list . The command \estrai[⟨*m*⟩]{⟨*list*⟩}{⟨*name*⟩} is the user command to extract the elements from the list {⟨*list*⟩} excluding m of them. The chosen elements will have the name \namei, \nameii, ... The elements of the list must be comma separated.

Example 17.
With the command \estrai[1]{sets,logic,powers}{alpha} you choose 2 elements of the set and these elements will have the name\alphai e \alphaii. If these names are names of exercises, it is possible to insert in the file list argument of the command \esercizi also \alphai e \alphaii.

\randestrai . The macro \randestrai[⟨*m*⟩]{⟨*list*⟩}{⟨*name*⟩} is similar to the macro \estrai, but the choice of the elements is completely randomized.

New vers. 2.1

\estraialfa The command \estraialfa{⟨*m*⟩}{⟨*list*⟩}{⟨*name*⟩} extracts {⟨*m*⟩} random objects from the list {⟨*list*⟩}, preserving the order. The chosen elements will have name \namei, \nameii, ... The elements of the list must be comma separated.

Example 18.
With the command `\estraialfa[2]{a,b,c,d}{alpha}` you choose 2 elements from the set, preserving the alphabetical order, and these elements will be named `\alphai` and `\alphaii`.

`\estraies` The command `\estraies[`⟨*m*⟩`]{`⟨*list*⟩`}` works as the command `\estrai`, but only on a list of exercises. Once the elements have been chosen, it prints them as the command `\esercizi`.

`\randestraies` The macro `\randestraies[`⟨*m*⟩`]{`⟨*list*⟩`}` works as the macro `\randestrai`, but only on a list of exercises. Once the elements have been chosen, it prints them as the command `\esercizi`.

8 The auxiliary files

8.1 The localisation file

The package has localisations in Italian (the default), UK English, US English, French, Spanish (thanks to Maria Hernandez Cifre), German (thanks to Kerstin von Kirschhausen), Serbian with latin alphabet (thanks to Dusko Latas), Greek (thanks to Sotiris Hasapis). Using greek language (and maybe other non latin localisations) it's necessary to load the packages `fontenc`, `inputenc` e `babel` before the package `esami`[7]. In this file one can also load some language dependent packages, such as `numprint`, `geometry`, etc. and there are the definitions of the commands `\shortdate` and `\longdate` to write the date.

N.B.: There is an known incompatibility between the package `babel` for the French language (which is not necessary for the localization) and the package `fp`: to manage this incompatibility, one can use the macro `\shorthandoff{:}` immediately after `\begin{document}`.

The `babel` package for the Spanish language makes a lot of characters active and this causes many incompatibilities (for instance in the `iffthen` package it is not possible to make number confrontations, sidely used in the `esami`) package. To solve this problem, you can use the *spanish* option when loading the article class, load the `babel` package with the option *spanish* and at least one among the options *es-minimal, es-sloppy* (for a complete list of the options and the possible incompatibilities, see the doc of the `babel-spanish` package[2].

8.2 The configuration file

The configuration file `esami-xyz.cfg` contains various definitions, i.e. the header (command `\testa`), the footer, the format for the solution file, the page style, the instructions for the exam.

8.3 The `master` and `master-sol` files

These two files differ only because the second one shows the solutions. They contain the instructions to generate the test. In both the files you must write the date (command `\date`) in the format YYYY/MM/DD, the number of versions to generate (command `\numcompiti`), and, obviously, the exercises. The test can be divided in parts and in each one of them you can use one or more of the environments and commands that we have presented.

The files has to be renamed before compiling them, otherwise compilation stops with an error message.

[7] The option *greek* requires at this moment the option *iso-8859-7* for the package `inputenc`

In the file, with the command `\maxLoopLimit`, the maximum number of tries that `\FPsetpar` should make in order to satsfy the conditions and how to calculate the initial seed for the generation of random numbers (command `\seme`).

It is also possible to define the name of the parts in which the test is divided.

8.4 The `totale-versioni` file

The file `totale-versioni` is used to generate all the variants of an exercise. With this file the package has always the option *prova* and you can use the option *param* to generate the parametric version of exercises.

Inside this file you should use only one command, `\def\esercizio{name}`, that should contain the name of the exercise to compile. If you compile the parametric version, the parameters and their intervals are shown at the end of the exercise.

9 Changelog

Version 1.0 (2013/03/08) First version released to CTAN.

Version 1.1 (2013/12/09) Inserted the dependance on the package `environ` to remove some errors in the LaTeX compilation. Removed the dependance on the package `icomma` in the italian localization file es-italian.lng because of the introduction of new features in the package `babel` for the Italian langaguage (macro `IntelligentComma`).

Version 2.0 (2015/02/25) Eliminated the dependance on `pstricks`. Modified the use of *fill-in* questions. Inserted the possibility to extract more than one variant from a file. Now it is possible to view the solutions in tests. Modified the visualization of decimal places in command `\FPsv`. Modified the definition of `\FPval` to allow the use of decimal places. Modified the algorithm of choice of a variant of an exercise, now completely deterministic. Added the command `\Acapo`. Added the serbian localization (latin alphabet). The compilation produces a new file with the list of the chosen exercises.

Versione 2.1 (2016/07/25) Modified `es-german.lng` and `es-spanish.lng` for an error in the input of the solutions. Introduced the macro `randestrai`. Introduced the macro `FPsignpol`. Introduced a control on the total points of the exercises. Solved the incompatibility with `babel` for the Spanish language. Removed the incompatibility between the command `\fillin` and the commands `\sempli` and `\semplix`.

Version 2.2 (2017/07/01) Introduced the possibility to have two different seeds for the choice of exercises (`\semeex`) and for the permutations and choice of the parameters (`\seme`): by default, they are equal. Slightly modified the command `\estrai` so that it is possible to use it to extract parameters from a list. Documentation integrated.

Version 2.3 (2017/09/18) Corrected a misprint in the code.

Version 2.4 (2018/02/10) Documentation corrected.

Version 2.5 (2018/05/06) Corrected a misprint in the code. Improved the selection of random exercises.

Version 2.6 (2021/10/11) Improved the spacing in `\FPsignpol`. Corrected the code to avoid some spourios spaces.

Version 2.7 (2021/10/11) Improved the spacing in `\sempli` and `\FPsignpol`. Corrected a misprint in the code

Versione 2.8 (2023/07/21) Documentation modified to solve a problem with the spanish language.

References

[1] Jason Alexander. The package `examdesign`. CTAN:/macros/latex/contrib/examdesign, 2006.

[2] Javier Bezos. Estilo spanish para el sistema babel. CTAN:/macros/latex/contrib/babel-contrib/spanish, 2021.

[3] Michael Mehlich. The package `fp`. CTAN:/macros/latex/contrib/fp, 1999.

[4] G. Messineo and S. Vassallo. The `esami` package for examinations. *TUGboat*, 34(1):40–46, 2013.

[5] Grazia Messineo and Salvatore Vassallo. Il pacchetto `esami` per la creazione di prove scritte. *ArsTeXnica*, 14:95–103, 2012.

[6] D. P. Story. The package `rangen`. http://www.math.uakron.edu/~dpstory/rangen.html, 2009.

[7] D. P. Story. Exerquiz & AcroTEX. http://www.acrotex.net/, 2012.

[8] Nicola L.C. Talbot. The package `probsoln`. CTAN:macros/latex/contrib/probsoln, 2011.