The examdesign class*

Jason Alexander

2001/3/26

Abstract

A new class, examdesign, for LaTeX 2ε is defined. It provides several features useful for designing tests or question sets: (1) it allows for an explicit markup of questions and answers; (2) the class will, at the user's request, automatically generate answer keys; (3) multiple versions of the same test can be generated automatically, with the ordering of questions within each section randomly permuted so as to minimize cheating; (4) the generated answer keys can be constructed either with or without the questions included; and (4) some environments are provided to assist in constructing the most common types of test question: matching, true/false, multiple-choice, fill-in-the-blank, and short answer/essay questions

1 Description

Teaching is a rewarding and enjoyable profession, designing exams is not. *Grading* exams is even worse, but for help with that last bit you'll need more than a LaTeX class. This class file started out with a relatively simply goal: randomize a collection of questions so that the author wouldn't need to do it by hand. Since then, it has grown in functionality and size and is now (I hope) a reasonably complete general-purpose class for constructing exams and question sets.

2 Usage

To begin, simply create a new LaTeX 2ε document starting with

```
\documentclass{examdesign}
\begin{document}
```

If you want to create an exam in 12pt type, use `\documentclass[12pt]{examdesign}` instead. If you need to load additional packages, they may be loaded via the command `\usepackage` as always. The class includes a number of packages included with a standard LaTeX distribution: keyval, multicol, and enumerate. If you don't have these packages, your LaTeX distribution is nonstandard (or incomplete), and you'll have to grab these packages from the nearest CTAN site in order to use examdesign.

Most exams/questions sets are split into sections according to the type of question asked, and examdesign follows this practice. There are five environments

*This file has version number v1.1, last revised 2001/3/26.

defined, one environment for each type of question: matching, shortanswer, truefalse, multiplechoice, and fillin. **N.B.** Previous versions of examdesign had starred forms of these environments of well. The starred forms (as well as the fixed environment, have been eliminated in this new release. Their functionality has been incorporated into the standard environments.

All the environments are designed to make binding answers to questions easier. This not only makes the job of writing the exam easier, but it allows the automated generation of answer keys.

2.1 Switches

There are two ways to customize the output of the exam class: switches and special formatting environments. Descriptions of the special formatting environments (such as frontmatter, endmatter, etc.) can be found in section 3.1.

A switch is a single command that changes an internal parameter influencing the overall formatting of the exam. Switches should be included in the preamble because doing so will guarantee that they will apply to the entire exam. Putting switches elsewhere may result in odd results because examdesign shuffles around what you write quite a lot, and you won't always know when a switch will be executed.

One change between this release and previous releases is that all of the class options unique to examdesign have been converted to switches for sake of consistency.

`\NoRearrange` Although the original reason behind writing examdesign was to automate the random rearranging of questions, people have pointed out that it would be nice to turn off the feature (say, when writing the exam). The `\NoRearrange` switch suppresses the rearrangement of questions. You can override this setting on a section-by-section basis, though, by setting the correct environment option (see the discussion of the options in section 2.2 for more details). In addition, see the discussion of the `\setrandomseed` macro below for how to generate "randomly rearranged" question sets in the *same* order across runs.

`\NoKey` `\OneKey` The switch `\NoKey` suppresses the generation of all answer keys, whereas `\OneKey` generates an answer key for the first exam only.

`\BoldfaceCorrectMultipleChoiceAnswer` `\UnderlineCorrectMultipleChoiceAnswer` The switch `\BoldfaceCorrectMultipleChoiceAnswer` identifies the correct answer in multiple choice questions by using boldface text instead of placing a box around the correct letter. The switch `\UnderlineCorrectMultipleChoiceAnswer` identifies the correct answer in multiple choice questions by underlining. These two switches are cumulative: including both in your exam will give you boldface underlined answers (the order doesn't matter). If neither one is specified, examdesign will default to boxing the letter of the correct answer. This is shown in the examples below.

Technical note: The ordinary LaTeX 2_ε `\underline` command does not work well when the underlined text spans several lines. Consequently, I've incorporated the ulem package, written by Donald Arseneau, for underlining. Since this package is embedded in this class, you do not need to load it again if you should need it.

`\NumberOfVersions` The `\NumberOfVersions` macro takes a single numerical argument—the number of different versions of the test to generate. Under the default settings, if you request more than 26 versions of the test you will get an error. This is because the forms are labelled according to letters of the alphabet, and LaTeX 2_ε doesn't like it when you try to ask for the 27th letter of the alphabet. If you *really* need

more than 27 copies of an exam, you will need to use one of the special formatting environments redefine the material inserted at the top of each exam and key.

`\class` The `\class` macro takes one argument: the name of the class to appear on the exam. For example, `\class{Math 205A: Set Theory}` will cause the name of the class to appear in the appropriate place (if you are using the default setting).

`\examname` `\examname` takes one argument, just like `\class`. The argument is used to identify the type of test. For example, `\examname{Final Exam}`, `\examname{Midterm Exam}`, `\examname{Pop Quiz}`,...

`\ConstantBlanks` `\ConstantBlanks` has one mandatory argument (a length). It tells TEX to typeset the underline in fill-in-the-blank environments using lines of a constant length, regardless of the length of the correct word.

`\ProportionalBlanks` `\ProportionalBlanks` has one mandatory argument (a real number r). It tells TEX to typeset the underline in fill-in-the-blank environments using lines that are exactly r times the length of the correct answer when the correct answer is typeset by TEX. The default setting is `\ProportionalBlanks{1}`. If you include both `\ConstantBlanks` and `\ProportionalBlanks` in the preamble, the exam will be typeset using whichever switch was last set.

`\StudentInfoLineSpacing` `\StudentInfoLineSpacing` requires a length as its argument. This value is used to determine the amount of additional spacing between lines in the student data box appearing in the upper right-hand corner of the exam. The default value is 6pt. If you change the definitions these environments using examtop and keytop, this command will have no effect.

`\SectionFont` This macro takes one argument, which can be any declarative font-changing commands (i.e., `\em` rather than `\emph`). It changes the default font used to typeset the section titles.

`\ContinuousNumbering` This macro takes no argument. It changes the default numbering behavior. Instead of resetting the numbers at the start of each section, it will continue the numbering from previous sections. (Notice that the matching and truefalse environments are not affected by this, as they don't number their questions.)

`\ShortKey` This macro takes no argument. It changes the way answer keys are typeset. Basically, it will delete any question text, any instructions at the start of a block of questions (or at the start of a section), and only print the answer.

`\DefineAnswerWrapper` This macro takes two arguments. For the shortanswer environment, the first argument will always be inserted before the answer, and the second argument will always be inserted after the answer.

`\SectionPrefix` This macro takes one argument, which specifies the default appearance of the section number preceding each title. Its default setting is
`\SectionPrefix{Section \arabic{sectionindex}. \space}`

2.1.1 Summary of Switches

Switch Name	Function
`\NoKey`	Suppress all answer keys
`\OneKey`	Generate only the first answer key, suppressing all others
\BoldfaceCorrectMultipleChoiceAnswer	See above
\UnderlineCorrectMultipleChoiceAnswer	See above
`\NumberOfVersions`	specify number of exams to make
`\class`	Class data (E.g., "Philosophy 29")
`\examname`	Exam name (E.g., "Midterm Exam")
`\ConstantBlanks`	See above
`\ProportionalBlanks`	See above
`\StudentInfoLineSpacing`	Set spacing in data box
`\SectionFont`	Change section font
`\ContinuousNumbering`	Change numbering style
`\ShortKey`	Change key formatting
`\DefineAnswerWrapper`	Change tags surrounding answers
`\SectionPrefix`	Change the appearance of the section number

2.2 Environments

There has been a significant change in the environments between this version of examdesign and earlier versions. In this version, all question environments can take an optional single argument which can be used to set a number of different parameters. (Before, the optional argument was used to set the title for a section.) Exams written for the previous version will have to be updated to work with the new version. Sorry.

The options available for each section are:

Name	Function
`title`	Specifies the title for each section. When no title is specified, no section number is displayed (giving an entirely blank header before the section). However, the section instructions will still be printed. If you would like a section number *without* a title, simply give a blank space for the title, or use `\relax`, as in `title={ }` or `title={\relax }`.
`rearrange`	This option takes one of two values, either `yes` or `no`. If `yes`,then the questions in that section are randomly rearranged; if `no`, then questions in that section appear in the order in which they are written. This option works for all environments (and hence replaces the old `fixed` environment).
`resetcounter`	This option takes one of two values, either `yes` or `no`. If `yes`, then the question numbers for that section start at 1, overriding the `\ContinuousNumbering` switch. If `no`, then the question numbers for that section will not be reset. This gives you section-level control over the numbering.
`keycolumns`	This option should be set to a number. It specifies the number of columns that should be used when typesetting the section in the answer key. For example, if you use the `\ShortKey` switch to only include the correct answer for multiple-choice questions, each answer takes up very little space on the line. If you have 40 questions, it would take 40 lines—a waste of paper. Setting `keycolumns=5` would typeset the answer key using 5 columns, requiring only 8 lines for the answers. The section instructions will *not* be typeset in columns, but will span the width of the page, instead. Note that it is very easy to get overfull/underfull box messages with this option (and the next).
`examcolumns`	Same as above, except it affects the exam.
`suppressprefix`	Can be set to either `yes` or `no`. If `yes`, it suppresses the section prefix for that particular section; if `no`, the prefix will appear as always. This option replaces the starred environments in the the previous release.

2.2.1 The fillin environment

This environment is for fill-in-the-blank questions. The syntax is:

```
\begin{fillin}[title={Insert title here}]
  \begin{question}
    How much \blank{wood} could a \blank{woodchuck} chuck if a woodchuck could
    \blank{chuck} wood?
  \end{question}
\end{fillin}
```

When typeset using `\ProportionalBlanks`, the questions will be printed in the exam as:

1. How much _____ could a _____ chuck if a woodchuck could _____ wood?

but in the answer key as:

1. How much wood could a woodchuck chuck if a woodchuck could chuck wood?

2.2.2 The shortanswer environment

The shortanswer environment does not format its questions in any way other than enumerating them. (It is then a catch-all environment for any other type of question.) It does allow the group of answers with questions. Including an answer is not mandatory (but you will get errors if you do not include an answer and ask for a `\ShortKey`). The general syntax is:

```
\begin{shortanswer}[title={Interesting questions...},
                    rearrange=yes]
  \begin{question}
    State Hobbes' definition of the state of nature and the role it plays in
    his social philosophy.
    \begin{answer}
      The state of nature is\ldots
    \end{answer}
  \end{question}

  \begin{question}
    State several examples that seem to illustrate that Mill's theory of
    Utilitarianism cannot be followed in practice, and explain why.
    Afterwards, state Mill's response to the problems you raised.
    \begin{answer}
      Often times it appears we need to act on our moral instincts without
      taking time to deliberate (as in the case of saving a drowning child).
      According to Mill, though, it seems that before each moral action we
      ought to deliberate in order to be sure that the action we take is such
      as to maximize the overall general utility\ldots
    \end{answer}
  \end{question}
\end{shortanswer}
```

Note that the answer environment appears inside the question environment.

In the exam, this will be typeset as:

1. State Hobbes' definition of the state of nature and the role it plays in his social philosophy.

2. State several examples that seem to illustrate that Mill's theory of Utilitarianism cannot be followed in practice, and explain why. Afterwards, state Mill's response to the problems you raised.

The answer key, in normal form with the default definition of `\DefineAnswerWrapper`, will appear as:

1. State Hobbes' definition of the state of nature and the role it plays in his social philosophy.

   Answer: The state of nature is...

2. State several examples that seem to illustrate that Mill's theory of Utilitarianism cannot be followed in practice, and explain why. Afterwards, state Mill's response to the problems you raised.

Answer: Often times it appears we need to act on our moral instincts without taking time to deliberate (as in the case of saving a drowning child). According to Mill, though, it seems that before each moral action we ought to deliberate in order to be sure that the action we take is such as to maximize the overall general utility...

If you specify a `\ShortKey`, only the answers will be presented:

1. **Answer:** The state of nature is...
2. **Answer:** Often times it appears we need to act on our moral instincts without taking time to deliberate (as in the case of saving a drowning child). According to Mill, though, it seems that before each moral action we ought to deliberate in order to be sure that the action we take is such as to maximize the overall general utility...

2.2.3 The truefalse environment

True/false questions. The general syntax is:

```
\begin{truefalse}
  \begin{question}
    \answer{False} Laden swallows fly faster than unladen swallows,
    especially if they carry coconuts.
  \end{question}

  \begin{question}
    \answer{True} Quantum mechanics was first stated using the
    mathematical tools of Hilbert spaces by John von~Neumann.
  \end{question}
\end{truefalse}
```

Again, note that the answer environment appears within the question environment. The above will be typeset in the exam as:

_____ Laden swallows fly faster than unladen swallows, especially if they carry coconuts.

_____ Quantum mechanics was first stated using the mathematical tools of Hilbert spaces by John von Neumann.

It will be typeset in the answer key as:

__False__ Laden swallows fly faster than unladen swallows, especially if they carry coconuts.

__True__ Quantum mechanics was first stated using the mathematical tools of Hilbert spaces by John von Neumann.

2.2.4 The multiplechoice Environment

It isn't difficult to format multiple choice questions in ordinary LaTeX with multiply nested enumerate's, but this environment provides for a simpler entry scheme, and the answer key can indicate the correct answer. The syntax is:

```
\begin{multiplechoice}
```

```
  \begin{question}
    How much wood could a woodchuck chuck if a woodchuck could chuck wood?
     \choice{A lot.}
     \choice{More than most.}
     \choice{Exactly $\pi$ cords.}
     \choice[!]{It depends on the nature of the woodchuck.}
  \end{question}
\end{multiplechoice}
```

In the exam, this is typeset as:

1. How much wood could a woodchuck chuck if a woodchuck could chuck wood?
   (a) A lot.
   (b) More than most.
   (c) Exactly π cords.
   (d) It depends on the nature of the woodchuck.

In the answer key, if neither of the options `mcbold` nor `mcunderline` were given, the answer will be typeset as:

1. How much wood could a woodchuck chuck if a woodchuck could chuck wood?
   (a) A lot.
   (b) More than most.
   (c) Exactly π cords.
   (d) It depends on the nature of the woodchuck.

You can have up to 26 choices. Also, notice that the correct answer was specified by giving the optional argument `[!]` to `\choice`.

2.2.5 The matching environment

This environment is new in this release of examdesign. It provides an environment for the creation of matching tests. The syntax is:

```
\begin{matching}[title={Some matching questions}]
  \pair{John Steinbeck}{\emph{The Grapes of Wrath}}
  \pair{Will Self}{\emph{My Kind of Fun}}
  \pair{Charles Darwin}{\emph{The Origin of Species}}
\end{matching}
```

In the exam, this will be typeset as:

_____ Will Self	(a) *The Origin of Species*
_____ Charles Darwin	(b) *The Grapes of Wrath*
_____ John Steinbeck	(c) *My Kind of Fun*

In the answer key, this will be typeset as:

(c)	Will Self	(a) *The Origin of Species*
(a)	Charles Darwin	(b) *The Grapes of Wrath*
(b)	John Steinbeck	(c) *My Kind of Fun*

The `\pair` macro preceeds the *matched* pair of statements. The environment will randomly shuffle the first and second columns in order to create the (hopefully randomized) lists. Because of the way the randomizer is implemented, this environment will produce better results (i.e., more scrambled) results the longer the list of questions is.

N.B. The block environment will break badly in this environment, so don't use it.

2.2.6 The block environment

The block environment lets you identify a group of questions, as well as some preceeding text, as a "block" that should be kept together even though the rest of the section gets rearranged. (The questions inside the block will not be rearranged. That's on my to-do list, but it will take more time than I have, and it's already been a long time since the last update.)

The syntax is straightforward. For example:

```
\begin{block}
Here is a spot where you can put some instructions, a graph, or whatever
you want to preceed the questions in this block.
 \begin{question}
   This is the first question.
   \begin{answer}
     This is the answer to the first question.
   \end{answer}
 \end{question}

 \begin{question}
   This is the second question.
   \begin{answer}
     This is the answer to the second question.
   \end{answer}
 \end{question}

 \begin{question}
   This is the third question.
   \begin{answer}
     This is the answer to the third question.
   \end{answer}
 \end{question}
\end{block}
```

Will keep those three questions together when the rearranging happens.

The block environment may be used in *any* of the previously described question environments, except for the matching environment. (There were some problems with using blocks inside some of the other environments with earlier versions—I believe those bugs have been fixed.)

The block environment can take a special optional argument, `questions`, that specifies the number of questions in the block. Specifying this option sets two counters, `first` and `last`, so that you can refer to the particular questions that fall within the block in the instructions. So, for example:

```
\begin{block}[questions=3]
\noindent For questions \thefirst--\thelast, consider the following data...
```

would, if the block appeared after question 13, be typeset as:

For questions 14–16, consider the following data...

In previous versions of examdesign, blocks were typeset indented from the left margin. This was undesireable because the block instructions could be misread as belonging to the end of the preceeding question. As of this version, block instructions are typeset using no such indent.

3 Special Formatting Tools

The following environments, macros, and length parameters allow the user to customize virtually every aspect of the exam. As I describe below, the default settings of the examdesign are defined using these environments.

3.1 Special Formatting Environments

`frontmatter` Any material enclosed in the frontmatter environment will be typeset on a page (or pages) by itself before the exam (and the key, too). For example, if one wanted to make a cover sheet for the exam with the class name and a place for the student to write their name, i.d. number, and so forth.

`endmatter` Any material enclosed in the endmatter environment will be typeset on a page (or pages) by itself after the exam (and the key, too). For example, if one wanted to place a page with several important equations, constants, and tables at the end of the exam for students to reference, if needed.

`examtop` Any material enclosed in the examtop environment will be typeset on the next page following the frontmatter text, if an exam is being created, and will be omitted if an answer key is being created. The default settings of the examdesign class use this environment to create the area for the class name, student name, etc. as follows:

```
\begin{examtop}
  \@@line{\parbox{3in}{\classdata \\
          \examtype, Form: \fbox{\textsf{\Alph{version}}}}}
          \hfill
          \parbox{3in}{\normalsize \namedata}}
          \bigskip
\end{examtop}
```

Where `\classdata` and `\examtype` are macros that use the current values assigned by the switches `\class` and `\examname`.

The definition of `\namedata` is:

```
\def\namedata{Name: \hrulefill \\[\namedata@vspace]
```

```
Student Number: \hrulefill \\[\namedata@vspace]
TA: \hrulefill \\[\namedata@vspace]
Date: \hrulefill}
```

Where `\namedata@vspace` is the length set by the switch `\StudentInfoLineSpacing`.

For those who don't know, `\@@line` is the LaTeX 2_ε equivalent of Plain TeX's `\line` command. If you wish to use `\@@line` in defining your own examtop or keytop, please be sure to use `\makeatletter` and `\makeatother` in the appropriate places, otherwise you could get errors.

`keytop` Any material enclosed in the keytop environment will be typeset on the next page following the frontmatter text, if a key is being created, and will be omitted if an exam is being created. The default settings of the examdesign class use this environment to create the top of an answer key as follows:

```
\begin{keytop}
\@@line{\hfill \Huge Answer Key
        for Exam \fbox{\textsf{\Alph{version}}}\fi \hfill}
        \bigskip
\end{keytop}
```

`exampreface` Any material enclosed in the exampreface environment will be typeset on the same page as the examtop text, right beneath it but before any sections of the exam are included. In case you are wondering what this environment does that cannot be done with examtop, your question is very good. The difference is this: The material enclosed in the exampreface environment is sensitive to whether the `twocolumn` option was given. If the `twocolumn` option *was* given, then the material in exampreface will be typeset at the start of the first column. By contrast, the material given in examtop is *always* typeset in a single column spanning the entire textwidth of the page, at the top of the exam.

`keypreface` Any material enclosed in the keypreface environment will be typeset on the same page as the keytop text, right beneath it but before any sections are included. As described above, if the `twocolumn` option *was* given, then the material in keypreface will be typeset at the start of the first column. By contrast, the material given in keytop is *always* typeset in a single column spanning the entire textwidth of the page, at the top of the key.

`examclosing` Essentially the same as the exampreface environment, with the obvious change that the enclosed material is typeset at the end of the exam, on the same page as the rest of the exam (if possible), but before the endmatter. (Any material enclosed in an endmatter environment is typeset on a page by itself after the exam or key.)

`keyclosing` Essentially the same as the keypreface environment, with the obvious change that the enclosed material is typeset at the end of the key, on the same page as the rest of the key (if possible), but before the endmatter.

3.2 Special Formatting Macros

Due to the special way the examdesign class constructs a document (see the technical notes in section 4), several special macros are provided which enable the user to easily take advantage of the way exams are constructed. In addition, there are several lengths which the user can modify to further customize the appearance of the exam.

`\exam` The `\exam` macro takes one argument, which can be any command or sequence

of commands subject to the constraints described in the technical notes. The argument will be included wherever it is if and only if an exam is being made, otherwise the argument will be ignored.

`\key` The `\key` macro takes one argument, which can be any command or sequence of commands subject to the constraints described in the technical notes. The argument will be included wherever it is if and only if a key is being made, otherwise the argument will be ignored.

`\examvspace` Just like the ordinary LaTeX command `\vspace`, with the exception that the vertical space is included if and only if an exam is being typeset.

`\examvspace*` Just like the ordinary LaTeX command `\vspace*`, with the exception that the vertical space is included if and only if an exam is being typeset.

`\examhspace` Just like the ordinary LaTeX command `\hspace`, with the exception that the horizontal space is included if and only if an exam is being typeset.

`\examhspace*` Just like the ordinary LaTeX command `\hspace*`, with the exception that the horizontal space is included if and only if an exam is being typeset.

`\keyvspace` Just like the ordinary LaTeX command `\hspace*`, with the exception that the horizontal space is included if and only if a key is being typeset.

`\keyvspace*` Just like the ordinary LaTeX command `\hspace*`, with the exception that the horizontal space is included if and only if a key is being typeset.

`\keyhspace` Just like the ordinary LaTeX command `\hspace*`, with the exception that the horizontal space is included if and only if a key is being typeset.

`\keyhspace*` Just like the ordinary LaTeX command `\hspace*`, with the exception that the horizontal space is included if and only if a key is being typeset.

`\word` The last new command (in version 1.02) is `\word`. This command allows the user to insert inessential changes in the wording of a question between various versions of the exam. Thus, not only will the order of the questions be different, but the actual *wording* of the questions will be slightly different as well. For example, if one includes:

```
\begin{question}
Is the most well-known \word{{astronomer} {wrestler} {physicist}} in
the world \word{{Carl Sagan} {Rowdy Roddy Piper} {Albert Einstein}}?
\end{question}
```

in an exam, the first version of the exam will include the question

1. Is the most well-known astronomer in the world Carl Sagan?

but the second version of the exam will include

1. Is the most well-known wrestler in the world Rowdy Roddy Piper?

and the third version of the exam will include

1. Is the most well-known physicist in the world Albert Einstein?

Let's call the argument of `\word` an *option-list*. If one requests more versions of an exam than options in an option-list, `\word` will behave as if the option list you gave "wrapped." That is, if one requests five copies of an exam but includes `\word{{A} {B}}` in the input file, `A` will be printed on the fifth version of the exam, because `\word` will act as if the option list was `\word{{A} {B} {A} {B} {A} {B}}`, where `A` is the fifth option.

If it turns out that your exams are becoming absolutely unreadable through excessive use of `\word`, chances are you are not introducing inessential changes in the wording of a question, but are actually trying to write two different exams simultaneously.

3.3 Length Parameters

The following table describes lengths that the user may set to whatever value they wish, and the effect that it has upon the appearance of the exam (or key).

Length	Initial Value	Effect
`\beforesectsep`	`\medskipamount`	Space inserted before the current section heading.
`\aftersectsep`	`\medskipamount`	Space inserted after the current section heading.
`\beforeinstsep`	`\medskipamount`	Space inserted before any instructions for the current section are typeset.
`\afterinstsep`	`\medskipamount-\topsep`	Space inserted after the instructions for the current section but before the questions.

Given the above parameters, there are two rules that determine whether space is inserted. (1) If the section heading is empty (as might happen if someone used a starred environment form and then did not include any optional argument), then *neither* `\beforesectsep` *nor* `\aftersecsep` are used to insert vertical space, and (2) if there are no instructions for a section, then *neither* `\beforeinstsep` *nor* `\afterinstsep` are used to insert vertical space.

4 Miscellaneous Goodies

`\setrandomseed` The `\setrandomseed` macro allows you to specify a random seed at the start of an exam. This allows you to have a "randomized" exam that keeps questions in the same scrambled order across different runs. If you set the random seed, it becomes possible to have crossreferences in the exam. In this release, the ordinary LaTeX commands `\label` and `\ref` work as expected, even across different forms of the test (with the questions in a different order).

Math environments using numbered equations should also work correctly in this version.

`\pagebreak` Having fixed the order of the questions, you might need to introduce a pagebreaks at certain points. In this release, the `\pagebreak` command works as expected, with one caveat. Putting a `\pagebreak` before a section will introduce a pagebreak before that section, and putting a `\pagebreak` before a question will introduce a pagebreak before that question—no matter where on the page that question may be.

`\BreakPageOnVersion` If you want to insert pagebreaks before questions or sections conditionally on the exam, you will need to use the `\BreakPageOnVersion{}` macro. This macro, which takes a number as its argument, will conditionally insert a `\pagebreak`

depending on whether number of the version currently being typeset equals the version given as an argument to the macro.

\IncludeFromFile
\InsertChunk
chunk

If you need to use any environment which changes the catcodes of the input characters (such as the verbatim environment, or XY-pic), you will have to employ the following hack. The reason for this has to do with the way examdesign scrambles the order of the questions. (Basically, each question is saved to a macro for later recall. This assigns each character a catcode when it is saved, so it is not possible to change the catcode at a later time.)

The hack is the following: create another file called "foo.tex" (you can name it whatever you like) and put `\IncludeFromFile{foo.tex}` in the preamble of your document. For each bit of text that you need contained category code changes (like the verbatim environment) put, in foo.tex, a named chunk containing that code. I.e.,

```
\begin{chunk}{chunk name}
\begin{verbatim}
  This is some verbatim text
  This is more verbatim text
\end{verbatim}
\end{chunk}
```

At the appropriate place in your exam, place `\InsertChunk{chunk name}`. This macro inserts the named chunk into the exam (or key) at that place. All catcode changing commands, environments, etc., should be capable of being inserted this way. (If you find one that doesn't, let me know.)

All of the chunks for a given exam can be stored in the same file—you just need to give each chunk a unique name. Only the chunk you request will be inserted at a particular point. If you have a lot of chunks, this can slow down considerably the overall processing of the exam, but there's no other general solution which will work.

The chunk names can be descriptive, but don't try anything too fancy. "Too fancy" meaning: if it causes an error, it's too fancy.

4.1 Bugs

No known bugs.

4.2 Changes

The changes between this version and previous versions are indicated in the above sections. Early releases of the examdesign class required that instructions for a given section be specified as such by using the `\instructions` command. The need for using `\instructions` has been lifted as of release 1.02. The rule is this: all material between the start of a section and the first `\begin{question}` or `\begin{block}` will be taken to be the instructions for that section. Old exams created with `\instructions` should still work with the new release (the `\instructions` macro has simply been redefined as `\relax`).

Thus, if you are using fixed to format a series of essay questions, you might want to tell students that they only need to write answers to two of the following three questions. The instruction text is always typeset at the beginning of a section, before any questions are typeset.