

AcroT_EX Software Development Team

**The eqexam Package
part of the
AcroT_EX eDucation Bundle**

D. P. Story

© 2005-2010 Version 1.7
March 12, 2010

<http://www.acrotex.net>
dpstory@acrotex.net

Table of Contents

1 Introduction	4
1.1 What's New	5
2 Required and Optional Packages	6
3 Installing eqexam	6
4 Building an Exam	6
4.1 The Preamble	6
4.2 The exam Environment	7
4.3 The problem and problem* Environments	9
• problem	9
• problem*	10
• Page Breaking	12
4.4 Special Constructs and other Gizmos	12
4.5 Fill-in Questions	13
• Short Fill-in Questions	13
• True/False Questions	13
• Long Fill-in Questions	14
4.6 Multiple Choice	15
4.7 Multiple Selection	16
4.8 Randomizing Choices	17
4.9 Gizmos and Gadgets	19
• The workarea Environment	19
• The \placeAtxy Command	20
• The splitsolution Environment	21
5 eqexam Options	22
5.1 Configuration Files	26
6 The online and email Options	27
6.1 The email Option	27
• Installing eqAttach.asp	29
• Setting up and Modifying the Script	30
• Some Options	30
• References	31
7 Bells, Whistles and other Customizations	31
7.1 Customizations	31
• Course Info Commands	32
• Changing the Title and Cover Page	33
• Changing the Running Headers	34
• Localization of Strings	35
7.2 Creating Multiple Versions of Exam	36

Table of Contents (cont.)

3

• The Original Version Scheme	37
• New Version Control	38
7.3 The Point and Totals Boxes	41
7.4 The eqComments Environment	42
7.5 The \OnBackOfPage Command	43
7.6 The \pushProblem and \popProblem Commands	44
7.7 \qNewPage and \aNewPage	45
7.8 Support for Solution Sets from a Text	45
7.9 Referencing Multiple Choice Questions	46
7.10 Displaying Points between two Markers	47
7.11 Extending the \fillin Command	48

1. Introduction

In my classroom work at The University of Akron, I've been using a personal \LaTeX package, which is called `eqexam`, for creating my in-class tests, quizzes, homework assignments, and review documents (pre-tests/sample tests). In recent weeks—at the end of the Fall Semester, 2004, and prior to the Spring Semester, 2005, I have filled the mundane and boring days with work on `eqexam`, fixing and enhancing it quite a bit.

The `eqexam` package is a stand-alone for \LaTeX , but is also tightly integrated with the \LaTeX eEducation Bundle. `eqexam` will be distributed by itself, as well as a part of the \LaTeX Bundle. The integration with the \LaTeX Bundle gives it many of the online features that users of the Bundle are familiar with.

Let's have an overview of the package, with suggestions for possible uses.

1. The first, and most obvious application of this package is to create a `pExam` or a `pQuiz`. (Here, the 'p' prefix refers to paper or pulp; thus, we can use `eqexam` to write paper Exams and/or pulp Quizzes). You can write the questions and the solutions, and publish (i.e., print the document on a printer) the exam/quiz with no solutions—ready to be taken in class—, or \LaTeX the source document with solutions listed after each question to create an answer key, for your personal use, or for the use by the class.
2. So much for pulp. Now on to 'e' (for electronic publication). In some of my classes, I put sample questions (review tests) on the web as PDF documents. In this case, you can create a PDF document without the solutions, and give the class time to solve the problems; then publish the document (in PDF on the web) with solutions. The solutions can appear immediately after the questions, or accumulated at the end of the document.
In the case where the solutions are at the end of the document, you can add links from the question to the solution.
Documents can be published with color (to enhance the on screen appearance) or can be published in black and white, meant to be printed. Or,...you can do both: a screen version and a paper version.
3. By invoking the `online` option, the white space left for hand-written answers to the questions become Acroform multi-line text fields, multiple choice questions become radio buttons, and fill-in questions also become text fields. The student can bring up the exam, and take it at a computer (in a CBT¹ lab). After the student is finished, he/she can print out the exam, and submit it to the instructor for traditional grading.
4. Now, here is an exciting feature of the `eqexam` package, that of email submittal! This feature is not too useful for technical fields (i.e., mathematics related fields) that require students to enter special symbols, but for some academic disciplines (English, History, Sociology, Politics and Government, etc.) this feature could be quite exciting.²

¹Computer Based Testing.

²Of course, I am addressing now the some six people worldwide in these fields that use \LaTeX and PDF! For you six, this feature is for you!

When you take the `email` option of `eqexam`, as with the `online` option, the white space left for hand-written answers to the questions become Acroform multi-line text fields, multiple choice questions become radio buttons, and fill-in questions also become text fields. Additionally, a button is automatically provided to submit by email the results of the test to the instructor. The results arrive at the instructor's mailer as an FDF attachment. The instructor can open the FDF and view in the originating PDF the responses given by the student.

The instructor can print out the document and grade in a traditional way, or if the instructor has **Acrobat Pro** or **Standard**, the instructor can use mark-up annotations within the PDF, save a copy of the students test to a class folder, and email a copy of the students exam, marked up with grade.³

If the exam is given for credit, it can be taken in a secure lab.

5. Perhaps a more reasonable application of this email submission feature of `eqexam` is the building and publication of surveys and questionnaires! Perhaps even teacher evaluations! The environments of `eqexam` can be easily used to write surveys and questionnaires to solicit the opinion of a target population. Responses are emailed to the designated person, who can summarize them.

By the way, speaking of summarizing results, a new feature of **Acrobat Pro 7.0**, allows you to take a folder of FDF files, such as the ones created by email submission, and extract all form fields and place results to a comma-delimited file (`.csv`). This comma-delimited file can be opened by a spreadsheet program and manipulated. Cool.

1.1. What's New

1. (Version 1.7) Added the ability to randomized items in a multiple choice/selection list. See 'Randomizing Choices' on page 17 and the `allowrandomize` option, as listed in 'eqexam Options' on page 22.
2. (Version 1.6) In this version, I've added the command `\this term` (see 'The Preamble' on page 6) and expanded the control of multiple versions, now you can have up to 26 versions of the same test! For details of this new multiple version scheme, see the discussion in 'New Version Control' on page 38.
Also added are `\forproblem` and `\foritem`, see 'Support for Solution Sets from a Text' on page 45.
3. (Version 1.4) Added a `manswers` environment for multiple choice questions where multiple selections are permitted. Exerquiz version 6.04 or greater required with the `online` and `email` options. See the section 'Multiple Selection' on page 16 for details.
4. (Version 1.3) Added in the `\bChoices` and `\eChoices` pair for specifying multiple choice alternatives. See the brief discussion in the section 'Multiple Choice' on page 15.

³Seems doubtful that anyone at this time has the expertise to do this! But it's available if anyone ever wants it.

2. Required and Optional Packages

The following packages that are not part of the normal \LaTeX distribution are *required*:

1. `calc`: Used for calculation of the position of the marginal points.
2. `pifont`: Used when the `proofread` option is used to indicate the correct answers to multiple choice questions.
3. `comment`: Used to have optional content, useful for developing exams for multiple sections of the same class.
4. `multicol`: Used to create questions in multi-column mode.
5. `verbatim`: Used to write solutions to the hard drive.

Additionally, the following packages may be used depending on the options chosen:

1. `web`: Used when the `pdf`, `links`, `online` or the `email` option is taken.
2. `exerquiz`: Used when the `links`, `online` or the `email` option is taken.

Of course, `web` and `exerquiz`, in turn, input a whole plethora of packages. Consult the documentation for the AcroTeX eDucation Bundle.

3. Installing eqexam

Create a folder in your `latex` search path named `eqexam` and place the package files `eqexam.dtx`, `eqexam.ins`, `eqexam.def` and any `.cfg` files. (If you have an `acrotex` folder, you can place the files there as well.)

Next, `latex eqexam.ins` to create `eqexam.sty` and `eqalone.def`. The other files (`*.tex` and `*.pdf`) can be placed anywhere.

The `eqexam` is a stand alone package that is tightly integrated with the AcroTeX Bundle. The file `eqexam.def` comes from the AcroTeX Bundle to provide the necessary support for many of the commands and environments defined in `eqexam`. The file `eqalone.def` are miscellaneous definitions that are needed for the stand-alone version. When you choose one of the options `links`, `online` or `email`, then `Exerquiz` is included in the package files. When you use one of these options you will need the most recent version of the AcroTeX eDucation Bundle, the one published concurrently with this package.

4. Building an Exam

In this section, we outline the steps to create an exam using the `eqexam` package, consult the sample exams for additional examples.

4.1. The Preamble

Of course, we begin with the standard `article` class, and the `eqexam` package:

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

The `<options>` are discussed in section 5. Next comes a exam identification information:

```

\title[T1]{Test 1}
\subject[C1]{Calculus I}
\author{D. P. Story}
\keywords{Calculus I, Section 004}
\university{%
    THE UNIVERSITY OF AKRON\
    Mathematics and Computer Science
}
\date{\thisterm, \the\year}
\duedate{October 17, 2005}

```

The `\title`, `\subject`, `\author` and `\date` are the same as is used in the web package. These are used by the standard \LaTeX macro to create the heading line of the first page of the exam, and are used in the running headers.

The `\title`, `\subject` have optional first arguments, where you can list a shortened version of the title or the subject. The shortened versions, if present, are used in the running headers.

The `\keywords` is used when you publish your exam in PDF and you use the `pdf` option (or `online`, `links`, `email`). The value of the argument of `\keywords` appears in the keywords field of the document info dialog.

When you take the `coverpage` option, the value of `\university` is used, along with some of the others on the cover page.

I've also defined a keyword of `\duedate`, this might be useful when using `eqexam` to create homework assignments with a due date, or just to record the date of the exam. The argument of `\duedate` fills the text macro `\theduedate`. So that if you say `\duedate{05/31/06}`, the macro `\theduedate` will expand to '05/31/06'.

Beginning with version 1.6, `\thisterm` is defined. The academic year of many American universities are divided into semesters (or terms); Fall, Spring, and Summer. The command `\thisterm` takes the current date and determines if it is the Fall, Spring or Summer Semester. For example, if the date of the compile is October 17, 2005, then `\thisterm`, `\the\year` expands to 'Fall, 2005'. This command is useful with the `\date`

The command `\thisterm` can be redefined to conform to the terms of the document author's university. See the definition in `eqexam.dtx`, copy and modify it.

4.2. The exam Environment

An exam is contained within the `exam` environment.

One of the things that I do in my courses, especially for the final exam, is to have a two-part exam. Typically, the first part is worth 100 points and covers the new material not already tested; the second part is usually a 50 point review. I grade these two parts separately and record them separately. Therefore, an `eqexam` test may contain one or more exam environments.⁴

After the preamble, we then say

```
\begin{document}
```

⁴Remember, this was originally a personal package, meant to suit my own needs.

```

\maketitle

\begin{exam}[Part I.]{Part1}

\begin{instructions}[Part I.]
Solve each of the problems without error. If you make an error,
points will be subtracted from your total score.
\end{instructions}
...
...
\end{exam}

\begin{exam}[Part II.]{Part2}

\begin{instructions}[Part II.]
The following is a short review of previously mastered material.
\end{instructions}
...
...
\end{exam}
\end{document}

```

After the `\begin{document}` and standard `\maketitle`, we begin an exam by opening an `exam` environment. This environment has two arguments the first optional, the second required. The first argument is a user friendly name (used when the solutions are listed at the end of the document); the second required argument is the name of the first part of the exam, `Part1` or `Part2`, for example. This argument is used to build the names of the PDF Acroform field names. This argument should consist of letters and numbers only.

Following the opening of the exam, typically, the instructor would have some instructions, this is the purpose of the `instructions` environment. It has one optional argument, the user friendly name of the part, say “Part I.”; if this optional parameter is not provided, then the word “Instructions.” is used. Following this label, the total number of points for this part is inserted, unless the `nosummarytotals` option is taken.

- ▶ The optional argument of the `instructions` environment has a color associated with it, and is visible when you compile the document with the `forcolorpaper` option. This color can be set by the command `\instructionsColor`; this command takes a single argument, a named color:

```
\instructionsColor{blue}
```

The above is the default definition.

At this point, you would insert your questions. Following the listing of all the questions (and optionally, their solutions), you finish up by closing out the `exam` environment.

Repeat, if additional parts to the exam are desired. Finally, finish off the document with `\end{document}`.

- ▶ You must `latex` your document *three times* to be sure all points have been properly calculated.

4.3. The `problem` and `problem*` Environments

All questions are posed using the `problem` and `problem*` environments. The former is for a single question, the latter is for a question with multiple parts.

- **`problem`**

The `problem` encloses a single question, the question itself may contain special constructs such as one or more fill-in the blanks.

The syntax for `problem` is

```
\begin{problem}[<num|empty>][h|H]
<Statement of question, which may contain special constructs>
...
...
\begin{solution}[<vspace>]
...
...
\end{solution}
\end{problem}
```

The environment takes two optional arguments. The first argument, `<num>` is the number of points for this problem, for example, if we want to have a 5 point question, we would begin the environment like so, `\begin{problem}[5]`; on the other hand, if we say `\begin{problem}`, the problem has no points associated with it.

The `problem` is actually a redefined `exercise` environment, as defined in `exerquiz`. The second parameter is inherited from the `exercise` environment. The second argument can optionally be an `h` or a `H`.

Use `h` if you do not want the solution to appear at the end of document (when you do not use the `nosolutions` or the `solutionsafter` options); the solution, however, will appear if the `solutionsafter` option is specified.

For the `H` argument, the solution will not appear at the end of the document (just as in `h`), nor will it appear if you specify the `solutionsafter` option.

To make things work correctly, if you do not want to have points for a question and want to hide the solution, use `[]` (empty brackets with no spaces) for the first argument.

```
\begin{problem}[][H]
($5$ Points Extra Credit) Solve this problem for extra credit.
\begin{solution}
This solution will not appear in all cases, unless the second
parameter is eliminated or is changed to h, in the latter case,
the solution appears just for solutionsafter.
\end{solution}
\end{problem}
```

Here, this problem has no points that will be added into the total number of points for the test.

The `solution` environment encloses the solutions. This environment is optional. The environment takes one optional parameter, namely the vertical space to leave for the student to work the problem. This vertical space is *created only* when the document author takes the `nosolutions` option. Thus,

```

\begin{problem}[10]
Do this problem.
\begin{solution}[2in]
This is the solution.
\end{solution}
\end{problem}

```

This defines a 10 point problem and leaves 2 inches of vertical space following the problem statement for the student to respond, provided the the `nosolutions` option has been taken.

- ▶ See ‘eqexam Options’ on page 22 for more details on the two options `nosolutions` and `solutionsafter`.

- **problem***

This environment is used when you want to ask a multi-part question, a series of related questions that are to be treated as a group.

The syntax is

```

\begin{problem*}[<num>|<<num>ea>|\auto|empty][\Do<num>]
Do each of the following problems, and be quick about it.
\begin{parts}

```

```

\item[h|H] The first question.
\begin{solution}[1.5in]
This is the solution to the first problem.
\end{solution}

```

```

\item[h|H] The first question.
\begin{solution}[3in]
This is the solution to the second problem.
\end{solution}

```

```

\end{parts}
\end{problem*}

```

The `problem*` environment takes two optional parameters, the first one takes one of four values:

<num> When the value of the first parameter is a number, this represents the total number of points for this multi-part question. Here, the instructor does not specify the weight of each part.

<num>ea When you specify a number followed by ‘ea’ (which is short for each). Thus, ‘[5ea]’ signifies that each part of this problem has weight of 5 points.

\auto If the value of the first parameter is `\auto`, then the total number of points is calculated automatically from the points defined by the `\PTs` macro. The `\PTs` would be placed following `\item` of each part that is to be given points. For example:

```

\begin{problem*}[\auto]
Do each of the following problems, and be quick about it.
\begin{parts}

```

```

\item\PTs{3} The first question.
\begin{solution}[1.5in]
This is the solution to the first problem.
\end{solution}

\item\PTs{4} The first question.
\begin{solution}[3in]
This is the solution to the second problem.
\end{solution}

\end{parts}
\end{problem*}

```

This defines a 7 point problem.

<empty> You need not specify any points at all, in this case do not include this first parameter, in which case, the second parameter is not used, so don't include it either.

Now for a description of the second parameter the `[\Do<num>]` parameter. In my senior- or graduate-level classes, I sometimes ask a questions with multiple parts. As part of the instructions for that problem I write, "Do exactly three of the following five problems." These questions are usually proof-type problems, and they can choose their best three to grade. In this context, all parts of the problem are of the same weight, that is, the `[<num>ea]`.

This is what `[\Do<num>]` does. When you specify `\Do3`, then only the points of 3 of the problems are added into the exam total. This second parameter is only checked if the first parameter is `[<num>ea]`. For example, specifying

```
\begin{problem*}[5ea][\Do3]
```

creates a 15 point question. By the way, this assumes there are 3 or more parts to this question.

By the way, there are two macros that are defined when the `\Do` is used, they are `\DoNum` and `\OutOfNum`; these expand to the (English) word for the number of problems to do, and the (English) word for the total number of problems. For example, if there were five parts to the problem below,...

```

\begin{problem*}[5ea][\Do3]
Solve exactly \textit{\DoNum} of the following {\OutOfNum}
problems. ....
\end{problem*}

```

The instructions would read, "Solve exactly *three* of the following five problems." These macros can be easily redefined to reflect other languages. The numbers themselves are contained in the two macros `\nDoNum` and `\nOutOfNum`.

- **parts** and **\item**: For a multi-part problem (`problem*`), the actual problems are enclosed in a `parts` environment, and each question is posed as an `\item` of that `list` environment. The command `\item` takes the `[h|H]` optional argument. As in the case of the `problem` environment, `h` prevents the solution from appearing at the end of the document (but it appears with `solutionsafter`), and `H` removes the solution in all cases.

- **Page Breaking**

The `exam`, `problem` and `problem*` environments use a (simple) page breaking algorithm to move a problem (or the beginning of an exam) to the next page.

If an `exam` environment begins at the lower third of the page, it is moved to the next page. You can influence this page break by using `\fvsizeskip` just before the beginning of the `exam` environment, like so,

```
\fvsizeskip{.4}
```

`\fvsizeskip` takes a decimal number between 0 and 1. In the example above, the environment will move to a new page if it begins in the lower $.4\text{\vsizes}$ of the page. The default value is $.3$.

There is a similar algorithm for `problem` and `problem` but is measured as a multiple of `\baselineskip`. If you place

```
\nbaselineskip{8}
```

just before a problem that appears near the bottom of the page, then it will be moved to the next page if it is within 8\baselineskip of the bottom. The default for this command is 6.

- ▶ Both `\fvsizeskip` and `\baselineskip` are one time commands, the default value is restored after the new value is read and used. The default values can be globally changed by redefining the following macros.

```
\def\default@fvsizeskip{.3}
\def\default@nbaselineskip{6}
```

The following are strategies for fitting the maximum number of questions on the minimum number of pages.

- 1. Moving:** Rearrange the order of the questions, if a problem can't fit entirely on a page, you can exchange move a shorter problem to that place, and move the longer problem to another page.
- 2. Tweaking:** Modify the space defined by the `solutions` environment to fit a problem on the page that is below it.
- 3. Placing work on back:** Using the `\OnBackOfPage` command, page 43, you can direct the student to answer the question on the back of another page, and thus, little space is needed to follow that question.
- 4. Working on separate sheets:** Of course, for some types of exams, the exam just contains the questions, the students answer the questions on separate sheets of paper. For this, you can use the `nospacetowork` option.

4.4. Special Constructs and other Gizmos

There are several useful commands for creating fill-in, true/false and multiple choice questions.

4.5. Fill-in Questions

In this section we cover the various fill-in constructs.

- **Short Fill-in Questions**

For a question requiring one or more short fill-in responses, eqexam has the `\fillin` command, the syntax is

```
\fillin[u|b]{<width>}{<answer>}
```

The first optional parameter is determined whether the fill-in is underlined ‘[u]’ or not ‘[b]’, the default it to underline the fill-in. The second is the amount of horizontal space you want to leave for the student to write in the response. The third argument is the correct answer. This correct answer will appear when you compile the document with the `answerkey` option.

- ▶ An example of `\fillin`.

```
\begin{problem}[5]
It is well known that \fillin{1in}{Newton} and \fillin{1in}{Leibniz}
are jointly credited as the founders of modern calculus.
\begin{solution}
It is well known that \underbar{Newton} and \underbar{Leibniz}
are jointly credited as the founders of modern calculus.
\end{solution}
\end{problem}
```

- ▶ When you choose the `online` or `email` option, `\fillin` generates a text field.

When the `usekv` option, and if the `xkeyval` package is available on the system, eqexam extends the capability and control of `\fillin`. See ‘Extending the `\fillin` Command’ on page 48.

- **True/False Questions**

True and false questions are, of course, just a special case of fill-in. A special command is available for true/false:

```
\TF{<answer>}
```

The single parameter is the correct answer (e.g., ‘T’ or ‘F’). The macro creates a underlined blank space `\defaultTFwidth` points wide (set to 30 points). The syntax for `\TF` is

```
\TF[<width>]{<answer>}
```

when `<width>` is not specified, `\defaultTFwidth` is used (and this value can be redefined).

The `\TF` command behaves differently from the generic `\fillin` command. Suppose you want to create a multi-part question (using `problem*`) consisting entirely of true/false questions. When an `\item` leads off with the `\TF` there are two possible formatting: This one...

- (a) _____ Isaac Newton is considered to be one of the founders of Calculus.

or this one...

(a) _____ Isaac Newton is considered to be one of the founders of _____ Calculus. The first alignment is the default, to get the second alignment, you need to set the value of `\fillinWidth` to the common width value of the `\TF` fields. For example,

```
\fillinWidth\defaultTFwidth
```

When `\fillinWidth` is set to a positive length (the common width of the `\TF` field), the second alignment above is created.

```
\begin{problem*}[3ea]
\textit{True} or \textit{False}.

\fillinWidth\defaultTFwidth

\begin{parts}

  \item \TF{T} It is well known that Isaac Newton and
  Gottfried Leibniz are jointly credited as the founders
  of modern calculus.
  ...
  \item ...
  ...
\end{parts}
\end{problem*}
```

- ▶ **Important:** The example above demonstrates the correct placement of `\fillinWidth`, just outside the `parts` environment, before it has the time to set up the paragraph shape of the environment.

The change is only local to that `parts` environment only. The `\fillinWidth` command goes outside a `problem*` environment, and can cause strange results if executed within a `parts` environment. Setting it to a `<width>` value other than the common width of the `\TF` fields will also create bad formatting.

- ▶ Just use `\fillinWidth` as illustrated in the above example.
- ▶ When you choose the `online` or `email` option, `\TF` generates a text field.

- **Long Fill-in Questions**

There is no special command for a longer response question, just leave enough vertical white space for the student to respond, for example,

```
\begin{problem}[5]
Do this problem
\begin{solution}[1.5in]
That's how you do it!
\end{solution}
\end{problem}
```

The above example leaves 1.5 inches of vertical space to do the work.

- ▶ When you choose the `online` or `email` option, this vertical space is changed into a multi-line text field.

4.6. Multiple Choice

For multiple choice questions, we use the `answers` environment. If the online or email option is taken, the choices are made into radio button fields so that *only one alternative* can be chosen. When multiple selections are permitted, the `answers` environment can be used, see ‘Multiple Selection’ on page 16.

```
\begin{problem*}[\auto] Answer each of the following.
\begin{parts} %\sqLinks
  \item\PTS{5} In what year did Columbus sail the ocean blue?
  \begin{answers}{6}
    \Ans0 1490 &\Ans0 1491 &\Ans1 1492 &\Ans0 1493
  \end{answers}
  \item\PTS{6} In what year did Columbus sail the ocean blue?
  \begin{answers}{1}
    \Ans0 1490
    \Ans0 1491
    \Ans1 1492
    \Ans0 1493
  \end{answers}
\end{parts}
\end{problem*}
```

Note: No solutions are given for this problem.

- Because the labels and values of the alternatives are based on the alphabet, the number of alternatives is restricted to twenty-six.

The `answers` environment is borrowed from `exerquiz`, operates the same way. The one argument is the number of columns to be used in displaying the alternative answers. If the number of columns is 1, a `list` environment is used, otherwise a `tabular` environment is used.

In the first item in the example above, we specify 6 columns, and must use tabular notation (separate columns with ‘&’) and end rows with ‘\’. The second item in the example above uses 1 column, the tabular notation is not needed, or used.

The `\Ans` macro is used to designate which alternative is the correct answer (1 for correct, 0 for not correct).

- Beginning with Version 1.3, an alternate style of specifying the alternatives is defined. A new pair of commands are defined: `\bChoices` and `\eChoices`. These two enclose the alternatives like so:

```
\begin{problem*}[\auto] Answer each of the following.
\begin{parts} %\sqLinks
  \item\PTS{5} In what year did Columbus sail the ocean blue?
  \begin{answers}{6}
    \bChoices
      \Ans0 1490\eAns
      \Ans0 1491\eAns
      \Ans1 1492\eAns
      \Ans0 1493\eAns
    \eChoices
  \end{answers}
  \item\PTS{6} In what year did Columbus sail the ocean blue?
  \begin{answers}{1}
```

```

\begin{bChoices
  \Ans0 1490\Ans
  \Ans0 1491\Ans
  \Ans1 1492\Ans
  \Ans0 1493\Ans
\end{bChoices}
\end{answers}
\end{parts}
\end{problem*}

```

Notice that the set of alternatives are the same, and are specified in exactly the same way; the first question, however, is a tabular environment with 6 columns (the argument of 6 of the `answers` environment), the second question is a list environment (since the argument `answers` environment is 1). Notice also that ‘&’ and ‘\’ are not used, and that each alternative is terminated by `\eAns`.

The `\bChoices` and `\eChoices` are creatures of the `exerquiz` package, and are fully documented in the reference for the `AcroTeX` Bundle, click on the link specifying “`AcroTeX` Bundle Documentation” on the first page.

- There are two styles of multiple choice: (1) enumerate the alternatives using letters; (2) enumerate the alternatives using boxes (that the student would check or fill-in). The default is (1), but you can change the default to (2) by using the `useForms` option. This styles can be locally changed by specifying the `\sqLinks` or `\sqForms` commands. In the above example, the `\sqLinks` command is commented out, but shows the correct position for it to change to style (1), which I am calling “links”. Within a multi-part, multiple choice set of questions, you can change one item to “links” and the next to “forms”, changes are local as long as you place the commands, `\sqLinks` or `\sqForms` within an environment (`parts`, `problem`, or `problem*`).

4.7. Multiple Selection

When writing a multiple choice question when more than one alternative is permitted, use the `manswers` environment (multiple answers). The distinction between the `answers` and `manswers` environments is lost when publishing to paper, but becomes important with the `online` and `email` options.

Use the `manswers` environment in the same way you use `answers`, except code in more than one correct answer. For example,

```

\begin{problem}[5]
Which of the following are primary colors?
\begin{manswers}{6} % specify tabular any with 6 columns
  \bChoices
    \Ans1 Blue\Ans
    \Ans0 Green\Ans
    \Ans1 Yellow\Ans
    \Ans0 Orange\Ans
    \Ans1 Red\Ans
  \eChoices
\end{manswers}
\begin{solution}
Yes, red, blue and yellow are primary colors.
\end{solution}
\end{problem}

```

You can use the `\bChoices/\eChoices` pair to specify the alternatives, or you can use the standard tabular notation. As with `answers` an argument of 1 specifies a list environment. See ‘Multiple Choice’ on page 15 for more examples on the use of the `\bChoices/\eChoices` pair.

4.8. Randomizing Choices

Beginning with version 1.7 of `eqexam`, the choices of a multiple choice/selection question can be randomized. The `random.tex` macro file by Donald Arseneau is used for this purpose.

The randomization is only allowed if the `allowrandomize` option of `eqexam` is used; otherwise, no randomization can occur.

The randomization is only defined for choices listed between the pair `\bChoices` and `\eChoices`. The `\eChoices` command now takes two optional key-value arguments:

- `nCols=<number>`: The number of columns to create, as described. You can also use the old style by specifying just `<number>`. Thus, `\bChoices[nCols=2]` and `\bChoices[2]` are equivalent.
- `random=<true|false>`: Specify this option if you want the choices to be randomized. You can use the key word `random` instead of `random=true`. For example, the following commands all will randomize the choices, `\bChoices[random]` or `\bChoices[nCols=2,random]` or `\bChoices[2,random]`. The default is to not randomize the choices.

The following is an example of the `random` option of `\bChoices`.

```
\begin{problem}[5]
Try to guess the correct answer.
\begin{answers}{3}
\bChoices[nCols=2,random]
\Ans0 1 a choice\eAns
\Ans1\label{eq} 2 another choice\eAns
\Ans0 3 still another choice\eAns
\Ans0 4 another\eAns
\Ans0 5 incoming\eAns
\Ans0 6 more choices\eAns
\Ans0 7 another still\eAns
\Ans0 8 too many\eAns
\Ans0 9 choices\eAns
\efreeze
\Ans0 10 None of these\eAns
\eChoices
\end{answers}
\end{problem}
```

Note the presence of the command `\efreeze`. Any of the items listed after `\efreeze` are not randomized, and are placed at the end of the list. So, for the example above, the first nine items will be randomized, whereas, the last item (None of these) will be placed at the end of the list.

Additional, there are five other commands that support the randomization feature.

```
\saveRandomSeed
\inputRandomSeed
```

A pseudo-random sequence of numbers requires an initial seed value. The `random.tex` macro file creates, by default, a seed value based on the data and time (the number of minutes since midnight); consequently, after every minute, the random sequence will change. By setting the value of the count register `\randomi`, the document author can also set the initial seed of the pseudo-random sequence.

The command `\saveRandomSeed` will write the last seed used in the source file to an auxiliary file (`\jobname_ran.sav`), while the command `\inputRandomSeed` inputs the seed stored in the `\jobname_ran.sav` back into the beginning of the source file. These two commands should be placed in the preamble.

By invoking both of these commands, a new pseudo-random sequence will be generated each time the source file is latexed.

Assuming a `\jobname_ran.sav` has already been created, by invoking the command `\inputRandomSeed` only (and not `\saveRandomSeed`), the seed already saved will be used for every subsequent compiling of the source document. Using the same seed is necessary in two situations:

1. When the document contains one or more `\label` commands, using the same seed gives you the same sequence every time you latex the document. This will give the auxiliary files a chance to come up to date so that any referencing of the label will be accurate.
2. When creating an exam with randomization that has several versions, which later you publish the solutions to, it is important that the randomization for the document is the same as that for the solution document. By using `\inputRandomSeed` (and not `\saveRandomSeed`), you should get the same sequence for the solution document (unless you modify the source file, adding or removing questions that have randomization).

Things to look for: If `eqexam` is not rearranging the order of the choices as you expect it to, it could be that `eqexam` is reading an old `.sav` file. Either delete that file in your source folder, or comment out `\inputRandomSeed` in your document.

```
\useRandomSeed{<number>}
```

You may have several sections of the same class take the exam with the questions rearranged for each. Save the seed value used by `eqexam` to randomized the choices (open the `.sav` and copy and paste line you see into your document, for example, it could read `\randomi=132088850`. Then use `\useRandomSeed` to use that seed value for that class, for example

```
\useRandomSeed{132088850} % 11:00 class
% \useRandomSeed{634952429} % 12:30 class
```

Of course comment out `\inputRandomSeed`.

```
\turnOnRandomize
\obeyLocalRandomize
```

The command `\turnOnRandomize` overrides all local settings of `\bChoices` and causes all choice lists to be randomized. While `\obeyLocalRandomize` returns control to the local settings. For example,

```
\turnOnRandomize
...
\bChoices
  \Ans... \eAns
  \Ans... \eAns
...
\eChoices
```

will cause the choice list to be randomized, even though the `random` option was not specified. Whereas, in this code

```
\turnOnRandomize
...
\obeyLocalRandomize
'''
\bChoices
  \Ans... \eAns
  \Ans... \eAns
...
\eChoices
```

the choices will not be randomized, because the `random` option was not specified; or they will be randomized if the `random` option is used.

Limitations: There are natural limitations on the use of `\bChoices` and `\eChoices` and consequently, there are limitations on the randomization. The content between `\Ans` and `\eAns` cannot have any verbatim text. This is usually not a problem for mathematical content, but could be a limitation for computer science where questions about syntax may be posed. I have in mind a work-around, but haven't pursued the problem as of yet.

4.9. Gizmos and Gadgets

I have a couple of crazy gizmos that you can use.

- **The `workarea` Environment**

For a mathematics test, we often pose a question that needs to be worked out. Vertical space is created by the `solutions` environment, and appears when the `nosolutions` option is used; however, often we want to mark up this vertical space with additional instructions, a diagram or a figure. The problem is how can the author write over the provided white space. For this, `eqexam` provides the `workarea` environment. The syntax is

```
\begin{workarea}[<width>]{<depth>}
...
```

```
Material that will overwrite the solutions vertical space.
...
\end{workarea}
```

This environment is placed immediately *after* the `solutions` environment, and the value of its parameter should be the same as the optional parameter of defined in `solutions`. The optional `<[width]>` parameter is the width of the work area, which is `\linewidth` by default. The required `<depth>` parameter is the depth of the work area, it should match the optional parameter of the `solutions` environment, directly above it.

```
\begin{problem}[3]
This is a question.

\begin{solution}[2in]
This is the solution, let's hope it's correct, or I would be
embarrassed to no end.
\end{solution}

\begin{workarea}{2in}
\textit{Hint}: Think long and hard before answering.
\par\vfill\hfill\setlength{\fboxsep}{2mm}
\fbox{Answer:\fillin[n]{1in}{The correct answer.}}
\end{workarea}
\end{problem}
```

When the `nosolutions` option is taken, the `solutions` leaves 2 inches of white space. The `workarea` environment that follows also specifies 2 inches, and the content of this environment will overlap the white space. (The student would then work around the written material.) Here, we give a hint, and leave an answer box (a fill-in) for the student to insert his/her answer.

When the `nosolutions` is not specified, the vertical space is not provided, and the `workarea` does nothing. If `solutionsafter` is specified, that space is replaced by the provided solution.

- **The `\placeAtxy` Command**

The `\placeAtxy` command is another device that I've used to place a block of text or a graphic on top of the vertical space created by the `solutions` environment with with the `nosolutions` option in effect.

```
\placeAtxy{<x_dim>}{<y_dim>}{<content>}
```

The first two arguments are the x and y coordinates (with dimensions) of where the `<content>` is to be placed. If this comment is placed below the `solutions` environment, then the origin is the lower left corner of the solutions box.

The following example, places the frame box Place a graph here (roughly) one inch up and one inch shifted to the right, from the bottom left corner of the `solutions` environment (when the `nosolutions` option is in effect). As with `workarea`, `\placeAtxy` does nothing if the `nosolutions` option has not been taken.

```

\begin{problem}[3]
This is a question.
\begin{solution}[2in]
This is the solution, let's hope it's correct, or I would be
embarrassed to no end.
\end{solution}
\placeAtxy{1in}{1in}{\framebox{Place a graph here}}
\end{problem}

```

The `\placeAtxy` command can also be used in combination with the `workarea` environment.

- **The `splitolution` Environment**

I developed this environment to solve a problem with the `online` and `email` options. The white space created by the `solutions` environment is converted into text fields (PDF form fields). If the `workarea` environment or the `\placeAtxy` command is used to place content on the white space, the student will be in the position of having to type on top of this content. (See the demo file `test01.tex` for an illustration of this.)

Therefore, it was necessary to have a way to separate the space reserved for the text field, and the additional content you might want to appear in this white space area. The `splitolution` environment is my solution to this problem.

- ▶ Consider the following example.

```

\begin{problem}[7]
This is a question worth $7$ points.
\begin{splitolution}{1.25in}
\begin{panel}[r]{1in}
\includegraphics[scale=.2]{fig1}
\end{panel}
\begin{solution}
This a really good solution. I hope this solution is correct or I
will be total embarrassed to no end. Even if it is wrong, maybe
the students will appreciate my tremendous effort. You can see
from the figure that the solution is obvious.
\end{solution}
\end{splitolution}
\end{problem}

```

After the statement of the problem comes the `splitolution` environment. It takes one required parameter, the length of the space to reserve vertically. (This is the same as the optional parameter of the `solutions` environment, as explained earlier.)

The `splitolution` environment *must* enclose two other environments: The `panel` and the `solutions` environments, *in that order*.

The `panel` environment comes first and takes one required and one optional argument. The required argument is a width dimension, its the width of the panel that will contain the material you want to write. (The depth will be the same as specified in the `splitolution` argument.) The optional parameter has takes a value of 'l' (the default) or 'r'. The l (resp., r) option means the panel is to appear on the left (resp., right) of the solution (or vertical white space).

After the `panel` environment comes the `solutions` environment. The optional parameter of this environment need not be specified, as it gets its value from the `splitsolution` parameter.

- The `splitsolution` environment creates a `.cut` file that contains the verbatim contents of the `panel` environment. This file is then later input. These `.cut` files can be deleted after you are satisfied with your document.

When the `nosolutions` option is *not* specified, there is a small gap of 3pt inserted between the panel and the solution. This the value of this gap is contained in the `\panelgap` command,

```
\newcommand\panelgap{3pt}
```

which can be redefined.

There is a save box `\eqpanelbox`, and two accompanying macros `\panelwidth` and `\panelheight` that can be used to measure the size of the panel.

- A variation on the previous example illustrates the usage of these three. The example puts a graphic into `\eqpanelbox` and then uses `\panelwidth` and `\panelheight` to set width and height.

```
\begin{problem}[5]
This is a question worth  $\$5$  points.

\abox{\eqpanelbox}{\includegraphics[scale=.2]{fig1}}

\begin{splitsolution}{\panelheight}
\begin{panel}{\panelwidth}
\includegraphics[scale=.2]{fig1}
\end{panel}
\begin{solution}
This a really good solution. I hope this solution is correct or I
will be total embarrassed to no end. Even if it is wrong, maybe
the students will appreciate my tremendous effort. You can see
from the figure that the solution is obvious.
\end{solution}
\end{splitsolution}
\end{problem}
```

Here, the graphic appears on the left.

- The depth you specify as the parameter of the `splitsolution` environment needs to be large enough to accommodate your typeset solution; otherwise, the solution will overlap the next problem. This is because, unlike the solutions inside a `solution` environment (but not in a `splitsolution` environment) are typeset in a minipage with a specified depth.

5. eqexam Options

The options documented here are entered as optional arguments of the `eqexam` package:

```
\usepackage[<optional_args>]{eqexam}
```

The optional arguments can also be introduced through `exambuilder.cfg`, the configuration file. Create a text file with the name of `exambuilder.cfg` and create the line shown below.

```
\ExecuteOptionsX{<optional_args>}
```

Place `exambuilder.cfg` in the folder of the source file and not on the \LaTeX search path.

- The `eqexam` package has numerous options, some inherited from `web`, some from `exerquiz`, and a number of new ones.

forpaper Take this option when you want to create a black and white paper version of your test.

forcolorpaper Take this option when you want to have a nice colorful paper version, or are publishing on the web in PDF.

nosolutions This is the normal option taken when you are printing a test for distribution to a class of students. When this option is taken, vertical space is generated by the `solutions` environment based on the value of its optional parameter. This leaves room for the student to solve/answer the question.

nohiddensolutions If you use the `h` optional parameter for `problem` or `\item`, the solution will not be listed (at the end of the document) *when you do not specify nosolutions*; but solutions will be typeset for the `solutionsafter` option. This option will override this feature.

noHiddensolutions If you use the `H` optional parameter for `problem` or `\item`, the solution will not be listed when you do not specify `nosolutions` or `solutionsafter`. This option will override this feature.

solutionsafter Causes solutions to appear following the statement of the problem.

preview The bounding boxes are shown when this option is taken, provided the `online` or `email` option is chosen. See the description of these two options below.

proofing Using this option will cause the correct answer for multiple choice questions to be marked with a check mark; the correct answers for fill-in questions (`\fillin` or `\TF`) are also shown.

The `answerkey` option, described below, executes the `proofing` and `solutionsafter` options.

- The following options are unique to the `eqexam` package.

pointsonleft The points for the problem are displayed in the left margin.

pointsonright The points for the problem are on the left margin.

pointsonboth Points are displayed in both margins.

nopoints Causes points not to be displayed, or calculated. Useful for writing documents that do not have points, such as a questionnaire.

totalsonleft The totals for each page can be displayed at the bottom left corner of each page using this option.

totalsonright The totals for each page can be displayed at the bottom right corner of each page using this option.

nototals Use this option if you don't want any totals at the bottom of the page.

noparttotals When using a test that has multiple `exam` environments, the totals since the last page are shown at the end of the environment along with a horizontal rule. This option turns off this feature.

There are two commands that can be used for local control of this feature, they are `\eoeTotalOff` and `\eoeTotalOn`. When an `exam` ends near the bottom of one page, the new `exam` will begin on the next page, this results in the horizontal rule being generated with the end of exam totals, and the totals at the bottom as well. If these two numbers are the same, then you can turn off the end of exam total using `\eoeTotalOff`. Use this command just above `\end{exam}` and the changes will be local to that exam part.

nosummarytotals When you use the `instructions` environment, the total points for exam are displayed following the instruction heading. Using this option turns off this feature.

coverpage Some instructors like to have a cover page for their exams, use this option to create a cover page. Use the `\eqexcoverpagedesign` command to design your own cover page.

nospacetowork When the `nosolutions` option is taken, the `solutions` environment leaves vertical space in which to respond to the question. Use this option to override this behavior.

The command `\SpaceToWork` causes the white space to be created again, and the `\NoSpaceToWork` turns it off again. Use these two commands to turn on and off the creation of vertical spaces in different parts of your exam.

answerkey This is a convenience option equivalent to `proofing` and `solutionsafter`. Useful for creating an “answer key” with answers and solutions displayed.

useforms Multiple choice questions have two forms, (1) the choices are labelled using letters (a), (b), (c), etc.; or (2) using a rectangular fill box. The default is (1). The `useforms` switches the default to (2). You can use the commands `\sqLinks` and `\sqForms` to change back and forth between these two types within the exam document. Using one of these commands outside a `problem` environment will globally change the default, from within, it will only change the default locally.

myconfig If this option is taken, eqexam looks for the configuration file eqexam.cfg. This configuration file is input at the end of the package, and can be used to redefine, for language localization purposes, any of the (text) macros described in this manual. See the section ‘Customizations’ on page 31 for a partial listing of macros that can be redefined and placed in eqexam.cfg.

myconfigi, myconfigii Two additional options, myconfigi and myconfigii are included. If you take one of these options, eqexam inputs either eqexam.i.cfg or eqexam.ii.cfg, respectfully. This gives you a total of three configuration files you can input, perhaps one for your exam format, another for your quiz format, and the third for your homework format. Does that cover it all?

obeylocalversions An option put in to give greater control over versions. Perhaps you have a eqexam file that has questions with multiple versions. You would like to pick and choose the versions to be used. In this case, using obeylocalversions will cause eqexam to obey any \selectVersion commands embedded in the document.

allowrandomize Use this option to randomize the multiple choice/selection questions. See ‘Randomizing Choices’ on page 17 for details.

showgrayletters When showgrayletters is used, multiple choice questions will have a gray capital letter A, B, C, etc. underneath it. This letter can then be referred to in the text or the solution using the \REF command.

See ‘Referencing Multiple Choice Questions’ on page 46 for more information.

usexkv When this option is used, and the document author has the xkeyval package on his/her system, there is a re-definition of the \fillin command. For more information, see ‘Extending the \fillin Command’ on page 48.

- ▶ The next two apply to files that have several versions in them, these were defined for use by the AeB Exam Builder utility, but they are available by the document author.

max This option takes a value; max=<N>, where N is a positive integer. The value of max is the number of versions for this document. This option executes numVersions{<N>} at the end of the package.

rendition This is a key-value pair. rendition=<alpha>, where alpha letter corresponding to the version that is to be typeset. At the end of the package, the command \forVersion{<alpha>} is executed.

- ▶ The next four options require the AcroTeX Bundle, and all of its required packages, such as hyperref, their use implies you are going to publish the document as a PDF.

pdf This option doesn’t do much, it brings in the web package, which in turn, places the values of the keywords (\title, \author, \subject, etc.) into the Document Description dialog of the PDF.

links This option brings in both `web` and `exerquiz`. When you do not use a solutions option (`nosolutions` and `solutionafter`), the solutions appear at the end of the document. When the `links` option is used, links from the questions to the solutions are created. Unless you use a “paper option” (`forpaper` and `forcolorpaper`), each solution is on a different page, making a document with a lot of pages. When you also specify a paper option, the solutions are separated by a `\medskip`.

online The `online` option implies the previous two options, but does more. When this option is taken, and the `nosolutions` option is specified, PDF forms are created: multiple choice questions become radio button fields; fill-in questions become text fields, and the vertical space created by the `solutions` environment become multi-line text fields.

This may be a useful option for an exam taken in a CBT⁵ lab, where the students can type in their responses and when finished, print the document to a lab printer to hand in.

email This option implies the `online` option, in addition, adds a submit button to the upper left corner of the first page of the exam. The student can take the test in a CBT lab, then submit the results to the instructor via email.

See the section ‘The `online` and `email` Options’ on page 27 for additional details of these last two options.

- ▶ When any one of the four options above are taken, a driver needs to be specified as well, the choices are...

dvipson For users of the Y&Y_{TeX} System, such as myself.

dvips For users of `dvips`, the dvi-to-postscript converter.

pdftex For users of `pdftex` application.

dvipdfm For users of `dvipdfm` application.

textures For textures users. (This option is totally untested.)

These options are passed on to `hyperref` and to `eforms`⁶ for the proper creation of links and form fields.

5.1. Configuration Files

The `eqexam` looks for two configuration files, they are `web.cfg` and `eqexam.cfg`.

The first one `web.cfg` may be already present on your hard drive if you use the Acro_{TeX} Bundle. Typically, desired default driver option is placed in here, for example, `web.cfg` might contain the single line,

⁵Computer Based Testing

⁶A component of Acro_{TeX} Bundle.

```
\ExecuteOptions{dvips}
```

for users of the dvips application for converting .dvi files to .ps file. The drivers supported by eqexam are listed in the previous section.

The second configuration file, eqexam.cfg, is input at the end of the package, provided the document author takes the myconfig option. Use this file to redefine some of the commands described in ‘Customizations’ on page 31, and elsewhere, to customize eqexam. An obvious use for this is to have a language customization of the package, input through eqexam.cfg.

If you place eqexam.cfg in the L^AT_EX search path, these customization will be global to all documents that specify the myconfig option. If it is placed in the source document folder (which is not in the L^AT_EX search path) the changes are local to all documents developed in that folder.

6. The online and email Options

When you use the online option, all fields created by the \fillin command, and this includes \TF, are converted into text fields, and the white space created by the solutions environment is converted to a multi-line text field. The fields manifest themselves when the document is viewed within the Adobe Reader, or any other PDF viewer that supports form fields.

This may be a useful option to the few people out there who are not in a technical field that requires specialized symbols to respond to a question. An exam created by the online option can be filled out online, printed, and submitted to the course instructor, perhaps within a lab setting.

There are other applications, such as creating a course survey, or a questionnaire of some type the students can fill out and submit. The email option may be more appropriate for these applications.

6.1. The email Option

When you pass the email option for eqexam, this does everything the online option does, in addition, it creates a “Submit” button that appears in the top-left margin of the exam (it does not appear on the cover page), and is placed there by the \maketitle command, that normally goes just after the opening of the document environment, \begin{document}.

The forms button is all setup to submit to the server-side script, eqAttach.asp, an active server page using vbscript as its scripting language. This script, when properly installed and functional, receives the form data generated by the document and attaches it to an email, which it sends off to the designated destination. Before discussing how to install and use eqAttach.asp, let me cover some commands that controls this button as well as options for changing what is sent to the server-side script.

When you take the email option, you need to supply a minimum of two pieces of information: the path to the server-side script eqAttach.asp and the email address of the person the results are to be sent. The command \SubmitInfo is used to supply this info, for example,

```
\SubmitInfo{http://localhost/scripts}{dpstory@uakron.edu}
```

This command takes two arguments, the first is the URL to the server-side folder that contains `eqAttach.asp`, the second argument is the email address of the recipient of the email. (You can have multiple recipients by separating the address by an comma.)

After the student submits the responses to the questions, an email is sent to the recipient (the instructor, perhaps). When the recipient receives the email, s/he can save the FDF attached file (containing the student responses) to a folder on the local hard drive. At least for a Windows machine when you open the FDF, the PDF will be fetched and the student data will be populated into the form fields.

Once this is done, the instructor can either save the populated file to the hard drive for later processing (the Acrobat application needed for this step) or print it to a printer for grading by hand.

If the instructor has Acrobat, s/he can use the markup capability of Acrobat to grade the electronic version of the test, and return the electronic version, with markup, to the student.

Below subject and message body of a “typical” submittal for the student “John Q. Student”.

Message Subject:

Exam Results: Test 1 of U. S. History

Message Body:

```
Exam Information:
  Course Name: U. S. History
  Exam: Test 1
  Student: John Q. Student
  TimeOfQuiz: 1/19/2005 12:07:56 PM
```

The FDF is attached.

The following commands can be used to modify the email message.

- ▶ `\EmailCourseName` is used to specify the name of the course. The default value for this is `\websubject`, obtained from the `\subject` macro used in the preamble; however, if you want a different name in the email, perhaps with more information included, you can redefine the value using this macro.

```
\EmailCourseName{\websubject} % the default
```

Important: When you use T_EX formatting in the subject, such as

```
\subject{\bfseries Calculus 1}
```

and you are using the `email` option, it will be necessary to use `\EmailCourseName` to redefine the subject, e.g., `\EmailCourseName{Calculus 1}`, to avoid possible T_EX compile errors, or to prevent T_EX primitives being a part of your email!

- ▶ `\EmailExamName` is used to specify the exam name of the course. The default value for this is `\webtitle`, obtained from the `\title` macro used in the preamble; however, if you want a different name in the email, perhaps with more information included, you can redefine the value using this macro.

`\EmailExamName{\webtitle}` % the default

Important: If you use some \TeX formatting in the title, such as

`\title{\bfseries Test 1}`

and you are using the `email` option, it will be necessary to use `\EmailExamName` to redefine the title, e.g., `\EmailExamName{Test 1}`, to avoid possible \TeX compile errors, or to prevent \TeX primitives being a part of your email!

- ▶ `\EmailSubject` The document author may want a custom subject in the email, instead of the standard one. By using this macro, he can design his own email subject.

`\EmailSubject{}` % the default

In this case `eqAttach.asp` inserts the standard one.

Exam Results: `\webtitle` of `\websubject`

The email would read “Exam Results: Test 1 of Calculus I”, for example.

To change the email subject we would put the following command in the preamble:

`\EmailSubject{Another Set of Cool Results}`

- ▶ `\ServerRetnMsg` The server script (`eqAttach.asp`) returns a message acknowledging the receipt of the data, this command allows the document author to customize the return message. The default definition is

`\ServerRetnMsg{}`

In this case `eqAttach.asp` inserts the standard one, “Exam results successfully sent to your instructor!”.

To change the return message to something more meaningful, put this command in the preamble, for example,

`\ServerRetnMsg{Your responses to the \TeX Survey have been received, thank you!}`

- ▶ `\SubmitButtonLabel` is the label that appears on the submit button.

`\SubmitButtonLabel{Submit}` % the default

- **Installing `eqAttach.asp`**

On the server side, in order for `eqAttach.asp` to run correctly, Microsoft Internet Information Server (IIS), version 4.0 or greater, is needed. The script `eqAttach.asp` needs to be placed where ASP scripts have execute permissions.

The `eqAttach.asp` uses the *Acrobat FDF Toolkit*⁷, version 6.0. Follow the directions for installation contained in the accompanying documentation.

Install `eqAttach.asp` in a folder (perhaps called `Scripts`) designated to execute scripts. If you don’t have such a folder, then the following steps explain how to create a virtual directory through IIS that points to this folder.

⁷Currently located at the Acrobat Family Developer Center.

1. Create a new folder on the system (Scripts, for example). Its recommended location is inside the Inetpub folder.
2. Place eqAttach.asp in this newly created folder.
3. In the MMC snap-in for IIS, create a virtual directory by right-clicking on the Default Web Site and selecting New > Virtual Directory.
4. Type "Scripts" (or whatever the name of the folder you created in Step 1) as the alias for the virtual directory, and then link it to the physical directory you created in Step 1.
5. Make sure that "Script execution" privileges are enabled. If not, enable them.

- **Setting up and Modifying the Script**

On the server side, in order for eqAttach.asp to run correctly, Microsoft Internet Information Server (IIS), version 4.0 or greater, is needed. The script eqAttach.asp should be placed where ASP scripts have execute permissions. There are two methods of sending e-mail:

1. CDONTS: This method (which is commented out by default) can be used on an NT server. Uncomment if you want to use CDONTS, and comment out the CDOSYS code lines that follow.
2. CDOSYS: This can be run on a Win2000 or WinXP server.

The script needs to be modified appropriate to your server, in particular, search down in eqAttach.asp for the configuration line

```
eqMail.Configuration.Fields.Item
    ("http://schemas.microsoft.com/cdo/configuration/smtpserver")
    = "mySMTP"
```

replace mySMTP with your SMTP server.

- **Some Options**

The default behavior of eqAttach.asp is to return a message to the document that indicates the receipt of the data, this message is "Exam results successfully sent to your instructor!" The message, as explained earlier, can be changed using the \ServerRetnMsg, like so

```
\ServerRetnMsg{Your TeX survey results have been received, thank you.}
```

Now, if for whatever reason you don't want this confirmation message to return to the document for display in alert box, you can sent the silent as part of the query string. For example, if

```
\SubmitInfo{http://myWebSite/scripts/eqAttach.asp?silent\#FDF}
    {myname@mymailprovider}
```

placed in the preamble of your document specifies the path to the script, silent mode, and the email address of the recipient of the form data.

Another other feature of `eqAttach.asp` that can be changed through the query string is the `/F` key-value pair of the FDF sent out in email. The value of this key is the path to the document that sent the FDF, it may be a url (an address on the Internet) or it could be a file specification of a local hard drive. If you specify `nopath` in the query string, like so

```
\SubmitInfo{http://myWebSite/scripts/eqAttach.asp?nopath\#FDF}
  {myname@mymailprovider}
```

then `eqAttach.asp` strips out the file path and leaves only the file name.

- ▶ This is what I did with the `tex_survey.tex` source file. I placed `tex_survey.pdf` in a LaTeX Survey folder on my desktop. As the emails came in, I saved the FDF attachments to this folder. By (double) clicking on the FDF, `tex_survey.pdf`, which is in the same folder, opened and the form data populated the fields from whence they were sent. It worked well for me.

If you don't use the `nopath` option, when you click on an FDF file you've received by email, your browser opens and the PDF on the Internet is brought into the browser and the form data populates the form fields, ...at least on a Windows machine. :-)

- **References**

The following links were used as a reference in the development of the `Email.asp` script.

- CDOSYS:
 - Invision Portal Tutorial: CDOSYS email tutorial
 - MSDN: CDO for Windows 2000. The IMessage Interface. (Use MIE to view this page.)
 - ASP 101 Sending Email Via an External SMTP Server Using CDO
- CDONTS
 - Juicy Studio The ASP CDONTS Component
 - DevASP Sending Mail from ASP with CDONTS.NewMail Object

7. Bells, Whistles and other Customizations

7.1. Customizations

We enumerate some commands for changing the default design of `eqexam`.

- **Course Info Commands**

`eqexam` has several commands for the student to provide some identification information.

- ▶ `\eqexamName`. This command defines the macro `\eq@ExamName` that creates the underlined space for the student to enter his/her name, and also defines the text box form field, in the case the `online` or `email` options are taken. There are two (design) parameters for `\eqexamName`

```
\eqexamName[<field appearance>]{<width>}
```

The first optional parameter can be used to modify the appearance of the text field, see the eForms documentation for details. The second parameter is the width of the field. The default definition is

```
\eqexamName[\Ff\FfRequired]{2.25in}
```

Here, the text field that will be generated (when `online` or `email` is specified) will be a required field. The total width of the space provided is 2.25 inches.

The command `\examNameLabel` controls the label to be used for this name field. It takes one parameter, the label to be used for the name field; the default definition is `\examNameLabel{Name:}`.

- ▶ `\eqSID`. This command defines the macro `\eq@SID` that creates the underlined space for the student to enter his/her student Identification number (SID), and also defines the text box form field, in the case the `online` or `email` options are taken. There are two (design) parameters for `\eqSID`

```
\newcommand\eqSID[<field appearance>]{<width>}
```

The first optional parameter can be used to modify the appearance of the text field, see the eForms documentation for details. The second parameter is the width of the field. The default definition is

```
\eqSID[\Ff\FfRequired]{2.25in}
```

Here, the text field that will be generated (when `online` or `email` is specified) will be a required field. The total width of the space provided is 2.25 inches.

The command `\examSIDLabel` controls the label used for this SID field. It takes one parameter, the label to be used for the name field; the default is `\examSIDLabel{SID:}`.

- ▶ `\eqEmail`. This command defines the macro `\eq@Email` that creates the underlined space for the student to enter his/her student email address, and also defines the text box form field, in the case the `online` or `email` options are taken. There are two (design) parameters for `\eqEmail`

```
\newcommand\eqEmail[<field appearance>]{<width>}
```

The first optional parameter can be used to modify the appearance of the text field, see the eForms documentation for details. The second parameter is the width of the field. The default definition is

```
\eqEmail{2.25in}
```

Here, the text field that will be generated (when `online` or `email` is specified). The total width of the space provided is 2.25 inches.

The command `\examEmailLabel` controls the label to be used for this email field. It takes one parameter, the label to be used for the name field; the default definition is `\examEmailLabel{Email:}`.

- **Changing the Title and Cover Page**

- ▶ `\maketitle`. The main heading that appears at the top of the first page of the exam is created by the \LaTeX (redefined) command `\maketitle`. The `\maketitle` has some code to place the email button in the top margin, followed by the expansion of the command `\maketitledesign`, whose definition is

```
\newcommand\maketitledesign
{%
  \makebox[\textwidth]{\normalsize
    \shortstack[l]{\strut\websubject\\@date}\hfill
    \shortstack[c]{\webtitle\\strut@altTitle}\hfill
    \shortstack[l]{\strut\eq@ExamName\\webauthor}}%
}
```

This command can be redefined using `\renewcommand` to suite your needs, for example,

```
\makeatletter
\renewcommand\maketitledesign
{%
  \makebox[\textwidth]{\normalsize
    \shortstack[l]{\strut\websubject\\webauthor, \@date}\hfill
    \shortstack[l]{\webtitle\\strut@altTitle}\hfill
    \shortstack[l]{\strut\eq@ExamName\\eq@SID}}%
}
\makeatother
```

This code adds in a field for the student to enter his/her student Id, here we enclose the code in a `\makeatletter/\makeatother` because this redefinition occurs in the preamble, and the code has an '@' in it.

Command elements that are appropriate to the redefinition are `\maketitledesign` are...

\websubject This is the course name, as determined by the `\subject` command.

\webtitle This is the exam name as determined by the `\title` command.

\altTitle An additional text field that is placed below `\webtitle`.

\@date This is the date as determined by the `\date` command.

\eq@ExamName This is the name field for the student to enter his/her name, as defined by default or redefined by `\eqexamName`, see 'Course Info Commands' on page 32.

\eq@SID This is the student ID field for the student to enter his/her ID, as defined by default, or redefined by the command `\eqSID`, see ‘Course Info Commands’ on page 32.

\eq@Email This is the student email field for the student to enter his/her email address, as defined by default, or redefined by `\eqEmail`, see ‘Course Info Commands’ on page 32.

\theduedate This is a text macro defined by the `\duedate` command. For example, setting `\duedate{03/10/05}` defines `\theduedate` so that it expands to 03/10/05. May be useful when redefining `\maketitledesign` for a homework assignment page.

- ▶ **\eqexcoverpagedesign**. When the `\coverpage` option is taken, a default cover page appears unless it is redefined. The `eqexam` package provides `\eqexcoverpagedesign` to design your own cover page. The default cover page uses the

\websubject This is the course name, as determined by the `\subject` command.

\webtitle This is the exam name as determined by the `\title` command

\webuniversity This is the value set by the `\university` command, given in the preamble.

\@date This is the date as determined by the `\date` command.

\eq@ExamName This is the name field for the student to enter his/her name, as defined by default or redefined by `\eqexamName`, see ‘Course Info Commands’ on page 32.

\eq@SID This is the student ID field for the student to enter his/her ID, as defined by default, or redefined by the command `\eqSID`, see ‘Course Info Commands’ on page 32.

\eq@Email This is the student email field for the student to enter his/her email address, as defined by default, or redefined by `\eqEmail`, see ‘Course Info Commands’ on page 32.

Copy the definition of `\eqexcoverpagedesign` from `eqexam.dtx` and modify as desired. Place the new definition in the preamble (enclosed between `\makeatletter` and `\makeatother`) or in a custom style file. No special support for this design is offered, because a cover page can be designed in so many different ways.

- **Changing the Running Headers**

There are two running headers, one header for the exam itself, and another when the solutions are shown at the end of the document.

- ▶ **Running Header for Exam**. The components of the running header occur on the left, center and right of each header, the commands `\lhead`, `\chead` and `\rhead`

1. `\lhead{<text>}`
Changes the left header text of the running header. This command defines an internal macro `\eq@lhead` that actually contains the text. The default is `\lhead{\shortwebsubject/\shortwebtitle}`
2. `\chead{<text>}`
Changes the center header text of the running header. This command defines an internal macro `\eq@chead` that actually contains the text. The default is `\chead{-- Page \arabic{page}\space of \eq@ExamLastPage\space--}`
3. `\rhead{<text>}`
Changes the right header text of the running header. This command defines an internal macro `\eq@rhead` that actually contains the text.
The default is `\rhead{\eq@ExamName}`.

If you want to redesign the layout of the running header, here is the macro that the above components fill.

```
\newcommand\runExamHeader{\eq@lhead\hfill\eq@chead\hfill\eq@rhead}
```

- **Running Header for Solutions.** The components of the running header for the solutions pages occur, as above, on the left, center and right of each header, the commands `\lheadSol`, `\cheadSol` and `\rheadSol`

1. `\lheadSol{<text>}`
Changes the left header text of the running header. This command defines an internal macro `\eq@lheadSol` that actually contains the text. The default is `\lheadSol{\shortwebsubject/\shortwebtitle}`
2. `\cheadSol{<text>}`
Changes the center header text of the running header. This command defines an internal macro `\eq@cheadSol` that actually contains the text. The default is `\cheadSol{-- Page \arabic{page}\space of \eq@ExamLastPage\space--}`
3. `\rheadSol{<text>}`
Changes the right header text of the running header. This command defines an internal macro `\eq@rheadSol` that actually contains the text. The default definition is `\rheadSol{SOLUTIONS}`.

If you want to redesign the layout of the running header, here is the macro that the above components fill.

```
\newcommand\runExamHeaderSol
{\eq@lheadSol\hfill\eq@cheadSol\hfill\eq@rheadSol}
```

- **Localization of Strings**

In this section we list various macros that expand to text appearing on an eqexam document. The default text is in English. These commands can be redefined to other English language phrases, or to other languages, and placed in the preamble of your document, or in one of the `.cfg` files.

- `\examNameLabel`: On each page of the exam, there is a place for the student to enter her/his name. `\examNameLabel` can be used to define the name label, the default is

```
\examNameLabel{Name:}
```

- `\ptLabel` and `\ptsLabel`: Labels for indicating the points of a problem, the first is the singular form of the second. The default is

```
\ptLabel{pt}           % singular form
\ptsLabel{pts}         % plural form
```

- `\eachLabel`: Label for indicating the common point value of each of several parts of the same problem.

```
\eachLabel{ea.}
```

- `\pointLabel` and `\pointsLabel`: The word for ‘points’ used in the instructions environment that lists the number of points in this exam. The default is

```
\pointLabel{point}     % singular form
\pointsLabel{points}   % plural form
```

The `\pointsLabel` command defines `\eq@pointsLabel`, which, in turn, is used in the `\summaryTotalsTxt`, the definition of which follows:

```
\newcommand{\summaryTotalsTxt}
  {(\summaryPointTotal\,\text{\eq@pointsLabel}$)}
```

- `\defaultInstructions`: The instructions environment has a default heading. The command `\defaultInstructions` allows you to change this heading. The default is

```
\defaultInstructions{Instructions.}
```

See ‘The Point and Totals Boxes’ on page 41 as well as the section ‘Course Info Commands’ on page 32 for additional details on these and commands useful for laying out the standard text of an eqexam document.

7.2. Creating Multiple Versions of Exam

Unfortunately, I teach multiple sections of the same course, and am faced with the problem of writing different exams for the same course each administered to a different section.

Typically, I only need for two variations on the test; however, further extensions can be made, if needed (See ‘New Version Control’ on page 38)

- **The Original Version Scheme**

The `eqexam` package defines a boolean switch, `\ifVersionA` for this purpose. The two sections of the same course are “Version A” and “Version B”. The default is that you are preparing an exam for “Version A”.

The command `\forVersion` sets the version: `\forVersion A` sets version to “Version A”, and `\forVersion B` set the version to “Version B”. (The argument of the `\forVersion` command is case insensitive, so you also type in `\forVersion b`.)

For small variations in text, there is the `\ifAB` macro,

```
\ifAB{<Version A text>}{<Version B text>}
```

for example, one could say,

```
\begin{problem}[2]
Compute $\frac{d}{dx}\ifAB{x^2}{x^3}$.
\end{problem}
```

For longer variations, the `comments` package is used to create comment environments that are included or excluded. The two environments are `verA` and `verB`.

```
\begin{problem}[2]
Compute $\frac{d}{dx}\ifAB{x^2}{x^3}$.

\begin{solution}[1in]
We use standard techniques:
\begin{verA}
$$
\frac{d}{dx} x^2 = 2x
$$
\end{verA}
\begin{verB}
$$
\frac{d}{dx} x^3 = 3x^2
$$
\end{verB}
\end{solution}
\end{problem}
```

There are several convenience macros for referring to the exams generated by the two variations.

Usually, an exam, test, homework assignment has a number associate with it, e.g. “Exam 1”, “Test 2”, “Assignment #12”, etc. This number should be defined using the `\examNum` macro.

```
\examNum{<number>}
```

where `<number>` is the number to be associated with the exam (test, assignment) under construction.

This command *must appear before* `\title` in the preamble. The command `\examNum` takes its argument and defines another macro `\nExam`, which has no arguments, but expands to `<number>`.

The `eqexam` package defines two commands `\Exam` and `\sExam` to automatically enter the test information for the current version. In the preamble, you can say,

```
\title[\sExam]{\Exam}
```

`\Exam` is the long version of the test name, and takes as its argument the exam number. `\sExam` is the short version, having no argument. For example,

```
\title[\sExam]{\Exam}
```

Both `\Exam` and `\sExam` use the value determined by `\examNum`, described above.

The text of `\Exam` and `\sExam` are generated by the four commands,

1. `\VersionAtext{<text>}` This is the text for the long version of the exam name for “Version A”. The default text is

```
\VersionAtext{Exam~\nExam--Version A}
```

2. `\VersionBtext{<text>}` This is the text for the long version of the exam name for “Version B”. The default text is

```
\VersionBtext{Exam~\nExam--Version B}
```

3. `\shortVersionAtext{<text>}` This is the text for the short version of the exam name for “Version A”. The default text is

```
\shortVersionAtext{Exam~\nExam A}
```

4. `\shortVersionBtext{<text>}` This is the text for the short version of the exam name for “Version B”. The default text is

```
\shortVersionBtext{Exam~\nExam B}
```

- All the above commands, 1-4, need to appear before `\title` in the preamble.

Below is a “typical” example of how to correctly redefine all the version text.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[forpaper,pointsonleft,noparttotals,nosolutions]{eqexam}

\examNum{1}
\forVersion{B}
\VersionAtext{Test~\nExam--003}
\VersionBtext{Test~\nExam--007}
\shortVersionAtext{T{\nExam}s3}
\shortVersionBtext{T{\nExam}s7}

\title[\sExam]{\Exam}
\author{D. P. Story}
\subject[C2]{Calculus II}
\date{Spring \the\year}
\keywords{Test \nExam, Section \ifAB{003}{007}}
```

- **New Version Control**

In this section we introduce a new set of commands that supersede the commands defined above. Those commands were limited to only two versions. The ones below can handle up to 26 versions.

The steps for creating a multiple version eqexam document are as follows.

```
\numVersions{<natural_number>}
```

In the preamble, declare the number of versions for this document using `\numVersions`, e.g., `\numVersions{3}`.

```
\longTitleText
  {<Text_1>}
  {<Text_2>}
  ...
  {<Text_n>}
\endlongTitleText
```

```
\shortTitleText
  {<Text_1>}
  {<Text_2>}
  ...
  {<Text_n>}
\endshortTitleText
```

Note: If there are more titles than what are declared, the rest of the titles are absorbed (gobbled). If there are fewer titles than declared, a \LaTeX package error is generated, and “fake” titles are generated.

Next, state the long and short titles for the document, one for each of the declared number of versions given earlier in `\numVersions`. For example, we can use the value `\nExam` in our titles. Usage:

```
\longTitleText
  {Test~\nExam--Version A}
  {Test~\nExam--Version B}
  {Test~\nExam--Make Up}
\endlongTitleText
\shortTitleText
  {T\nExam A}
  {T\nExam B}
  {T\nExam MU}
\endshortTitleText
```

These two commands give values to `\Exam` and `\sExam`. If `\forVersion{a}` is executed, `\Exam` expands to the text `Test~\nExam--Version A` and `\sExam` expands to `T\nExam A`, using the example above. The value of `\nExam` is determined by the `\examNum` command, as described above.

Next is the command that does all the work. It creates alternate text macros for each of the versions declared using `\numVersions`. The syntax is

```
\forVersion{<letter>}
```

For example, assuming `\numVersions{3}`, `\forVersion{a}` (or `\forVersion{A}`) defines 3 text commands `\vA`, `\vB` and `\vC`, each taking one argument, the text you want to display:

Name the $\backslash\text{vA}\{\text{place}\}\backslash\text{vB}\{\text{date}\}\backslash\text{vC}\{\text{year}\}$ of the signing of the Magna Carta.

Since $\text{forVersion}\{a\}$ was declared, only the $\backslash\text{vA}$ text is displayed, the others are gobbled up. But wait, time out, the $\backslash\text{forVersion}$ does more than that! It also creates a series of comment environments $\backslash\text{begin}\{\text{verA}\}/\backslash\text{end}\{\text{verA}\}$, $\backslash\text{begin}\{\text{verB}\}/\backslash\text{end}\{\text{verB}\}$, $\backslash\text{begin}\{\text{verC}\}/\backslash\text{end}\{\text{verC}\}$, etc., where only the version for which this compile applies will be typeset, the others are commented out.

```
\numVersions{3}
\forVersion{b}
...
\begin{document}
...
Solve the equation for  $\backslash\text{vA}\{x\}\backslash\text{vB}\{y\}\backslash\text{vC}\{z\}$ :
\[
\begin{verA}
2x + 4 = 7
\end{verA}
\begin{verB}
5y + 2 = 4
\end{verB}
\begin{verC}
3z - 2 = 2
\end{verC}
\]
```

Here is a final example of the multiple version scheme, taken from the preamble of one of my Calculus tests.

```
\documentclass{article}
\usepackage{amsmath,graphicx}
\usepackage[forpaper,pointsonleft,nototals,nosolutions]{eqexam}
%\usepackage[forpaper,pointsonleft,nototals,answerkey]{eqexam}

\numVersions{3}
\forVersion{a}
\examNum{1}
\longTitleText
{Test \nExam--Version A}
{Test \nExam--Version B}
{Test \nExam--Make Up}
\endlongTitleText
\shortTitleText
{T\nExam A}
{T\nExam B}
{T\nExam MU}
\endshortTitleText

\subject[C3]{Calculus III}
\title[\sExam]{\Exam}
\author{Dr.\ D. P. Story}
\university
{%
    THE UNIVERSITY OF AKRON\
    Department of Theoretical and Applied Mathematics
```

```

}
\date{\thisterm\space\the\year} % Fall 2005
\duedate{09/26/05} % actual date of the test
% If you convert to pdf using a pdf (links, online, email)
% option, this will appear in the keywords field of the
% document info dialog.
\keywords{\Exam, administered \theduedate}

```

There is one additional command that can be used to locally control which version that is typeset in the document.

```
\selectVersion{<number>}{<total_versions>}
```

You can place the `\selectVersion` command in front of a question or a part of a question that has multiple versions. Through this command you can select which version to typeset, provided the option `obeylocalversions` is set. For example,

```

\selectVersion{3}{4}
\begin{problem}[10]
...
\end{problem}

```

The `\selectVersion` command says there are four variations on the next question and the document author wants to use the third one (that would correspond to C, in the `\forVersion` command). Again, the `obeylocalversions` must be taken for `eqexam` to obey this command.

Recommendation: I recommend that each problem have the command `\selectVersion` in front of it, even for parts. Suppose the document author says `\numVersions{5}`, but some problems don't have five versions, what do you do? If there is a `\selectVersion` in front of a problem with multiple versions, the `\selectVersion` will partially expand to determine if it is needed. It is needed if the version specified by `\forVersion`, is greater than the number of versions for the problem. In this case, `\selectVersion` performs modular arithmetic to compute which version is to be used. For example, if `\forVersion{E}` has been declared in the preamble, but a problem has only three variations, the `eqexam` will use variation B; if `\forVersion{D}` was declared, version A is used, and so on.

7.3. The Point and Totals Boxes

There are two types of points boxes, and only one type of totals box. All the commands listed below can be redefined for language localizations, for example.

- Points that appear in the left margin (the `pointsonleft` or `pointsonboth` options). There are two text macros that are used,

```
\newcommand\leftmarginPtsTxt[1]{\small$#1^{\text{pts}}$}
```

when the total points for that problem are shown, and the other

```

\newcommand\leftmarginPtsEaTxt[1]
{\small$#1_{\text{ea.}}^{\text{pts}}$}

```

when the document author indicates that each sub-part of a problem is weighted the same, (when the author begins a `problem*` environment with `\begin{problem*}[3ea]`, for example).

- Points that appear in the right margin (options `pointsonright` or `pointsonboth`). These points appear in the bottom half of a box, the text for that box is determined by the following definition.

```
\newcommand\marginpointsboxtext[2]{\small\#1\,\text{pts}}}
```

By the way, the purpose of the upper part of the box is for the instructor to enter the number of points a student received for that problem.

- Points specified by the `\PTs` command. This text is defined by `\itemPTsTxt` as follows. See the paragraphs on ‘`problem*`’ on page 10 for a discussion of the use of `\PTs`.

```
\newcommand\itemPTsTxt[1]{(\#1$ pts)}
```

- The totals box. When you specify either option `totalsonleft` or `totalsonright`, you get a page totals box appearing in the lower left or right bottom corner.

```
\newcommand\totalstext{\small$\theeqpointsthispage\,\text{pts}}}
```

where `eqpointsthispage` is a counter whose value at the end of each page *should* be the page total. For tests that have multiple exam environments, if one exam part ends on a page, and another begins on the same page, this number (`eqpointsthispage`) is the total on the page from the beginning of the new exam part. In this case, at the end of the exam part, there should also appear a remaining total for that part on that page.

- Summary Totals. When you use the `instructions` environment to give initial instructions for an exam, the total points appears automatically in the text, unless you specify the `nosummarytotals` option. This text is defined by `\summaryTotalsTxt`, whose definition follows:

```
\newcommand\summaryTotalsTxt{(\summaryPointTotal\,\text{points}}}
```

where `\summaryPointTotal` is a macro that expands to the total for this exam environment.

7.4. The `eqComments` Environment

In addition to the `instructions` environment, as explained in the section 4.2, entitled ‘The exam Environment’ on page 7, should you want to insert additional instructions from within the body or the exam, use the `eqComments` environment. The `eqComments` environment has one optional argument, a formatted heading for the comments you want to make. For example,

```
\begin{eqComments}[Proofs.]
Solve each of the problems 5--8 on a separate sheet of paper,
do not write on the back of the paper. Follow the instructions
provided for each problem. Use your little gray cells.
\end{eqComments}
```

Such instructions must go between problems, of course, not within the body of either a `problem` or a `problem*` environment.

- The optional argument has a color associated with it, and is visible when you compile the document with the `forcolorpaper` option. This color can be set by the command `\eqCommentsColor`; this command takes a single argument, a named color:

```
\eqCommentsColor{blue}
```

The above is the default definition.

7.5. The `\OnBackOfPage` Command

In order to reduce the number of pages needed for an exam, I often cheat by asking the student to work on the back of one of the test pages.

```
\newcommand\bopText{on the back of page~\boPage}
\newcommand\bopCoverPageText{(the cover page)}
\newcommand\OnBackOfPage[1][\bopText]{%
```

For this, I use the the `\OnBackOfPage` command

```
\OnBackOfPage[<optional text>]
```

The optional argument allows you to enter variational text, text that varies from the default text. The default text is contained in `\bopText` macro, its definition is

```
\newcommand\bopText{on the back of page~\boPage}
```

where `\boPage` is the page the student is instructed to do the work. Thus, if you say, “Continue `\OnBackOfPage`.” This would expand to “Continue on the back of page 2.”, or whatever `\boPage` is determined to be.

To illustrate the use of the optional argument of `\OnBackOfPage`, you might say,

```
\OnBackOfPage[The back of page~\boPage] can be used to continue work,
if necessary.
```

This expands to “The back of page 2 can be used to continue work, if necessary.”

The algorithm used to compute the page, `\boPage`, on which to continue to work is as follows: For all pages, except for the first page of the test, the student works on the back of the previous page. For the first page of the test, the student works on the back of the first page, unless there is a cover page, in which case the student is instructed to work on the back of that page.

In the case of working on the back of the cover page, there is a variation on the instructions, `\OnBackOfPage` expands to “on the back of page 1 (the cover page)”. The phrase “(the cover page)” can be redefined using the `\bopCoverPageText` command. The definition of this command is

```
\newcommand\bopCoverPageText{\space(the cover page)}
```

We could change this as follows,

```
\renewcommand\bopCoverPageText{, the cover page}
```

so that it would now read, “on the back of page 1, the cover page”. To remove this feature altogether, you could redefine as

```
\renewcommand\bopCoverPageText{}
```

7.6. The `\pushProblem` and `\popProblem` Commands

There may be an occasion when a multi-part question needs to be broken between parts. use the `\pushProblem` and `\popProblem` for this purpose. The push saves the counter value, and ends the `parts` environment. The pop restarts the `parts`, and resets the `parts` counter.

In the `multicols` environment below, we `\pushProblem`, then close `multicols`, we execute `\popProblem`, and then continue with the multi-parts in single column.

```
\begin{problem*}[\auto]
Do each of the following without error.
\begin{multicols}{2}
  \begin{parts}
    \item\PTS{3} This is a problem.
    \begin{solution}[1in]\end{solution}

    \item\PTS{3} This is a problem.
    \begin{solution}[1in]\end{solution}
  \pushProblem
\end{multicols}
\popProblem
  \item\PTS{4} Do this harder problem.
  \begin{solution}[.5in]\end{solution}
\end{parts}
\end{problem*}
```

In the example, the first two questions appear in two column format, while the third appears in single column format. The same thing can be done in reverse, like so:

```
\begin{problem*}[\auto]
Do each of the following without error.
  \begin{parts}
    \item\PTS{3} This is a problem.
    \begin{solution}[1in]\end{solution}
  \pushProblem
\begin{multicols}{2}
\popProblem
  \item\PTS{4} This is a hard problem.
  \begin{solution}[1in]\end{solution}
  \item\PTS{4} Do this harder problem.
  \begin{solution}[1in]\end{solution}
\end{parts}
\end{multicols}
\end{problem*}
```

Now, first question is in single column and the next two are in two column format.

- In order to get the correct formatting, the `multicol` environment must begin before the `parts` environment.

See `quiz02.tex` for an example of `\pushProblem` and `\popProblem`.

7.7. `\qNewPage` and `\aNewPage`

The command `\qNewPage` (questions newpage) and `\aNewPage` (answers newpage) are convenience commands for creating new pages. The first one expands to `\newpage` when the `\ifanswerkey` is false and the second one expands to `\newpage` when the `\ifanswerkey` is true; their definitions are

```
\newcommand\qNewPage{\ifanswerkey\else\newpage\fi}
\newcommand\aNewPage{\ifanswerkey\newpage\fi}
```

7.8. Support for Solution Sets from a Text

I use `eqexam` not only for exams, quizzes and homework assignments, but also for solution sets for problems assigned from the text.

Suppose the assignment was to solve, on a certain page in the text, problems which include **2**, **6** and **12(b)(d)** and it is desired to provide solutions to these problem. For this purpose, `eqexam` provides `\forproblem` and `\foritem`. These commands are used as follows:

```
\begin{exam}{HW\nExam}
\begin{instructions}[Description] (10 points)
Selected solutions from Assignments~24, 25, \S7.1.
\end{instructions}
```

```
\begin{eqComments}[\S7.1]
\textbf{Solving Linear Equations}
\end{eqComments}
```

```
\forproblem{2}
\begin{problem}
Statement of problem.
\begin{solution}
Solution to this problem.
\end{solution}
\end{problem}
```

```
\forproblem{6}
\begin{problem}
Statement of problem.
\begin{solution}
Solution to this problem.
\end{solution}
\end{problem}
```

```
\forproblem{12}
\begin{problem*}
Statement of problem.
\begin{parts}
\foritem{b} Statement for item (b)
\begin{solution}
```

```

        Solution to this problem.
    \end{solution}

    \foritem{d} Statement for item (d)
    \begin{solution}
        Solution to this problem.
    \end{solution}
\end{parts}
\end{problem*}
\end{exam}

```

7.9. Referencing Multiple Choice Questions

When the `showgrayletters` option is used, each alternatives in a multiple choice question will have a gray capital letter A, B, C, etc. underneath it. This letter can then be referred to in the text or the solution.

The use of this option is global and is controlled by the switch, `\ifaebshowgrayletters`. The gray letter feature can be turned on and off locally: To turn on this feature, insert the command `\graylettersOn` at some appropriately chosen point in the document; to turn off the gray letter feature insert `\graylettersOff`.

In support of the `showgrayletters` option is a new command `\REF`. `\REF` acts like the \TeX command `\ref` with the `hyperref` modifications, but it converts the reference to uppercase. When `\ref` would typeset the letter ‘a’, for example, `\REF` would typeset the letter ‘A’. `\REF`, like `\ref`, typesets a `hyperref` link. `Hyperref` defines a `*` version of `\ref`; `\ref*` typesets the reference, but does not create a link; `\REF*` does the same. When `\aebshowgraylettersfalse` is in effect, `\REF` does not capitalize the reference.

- Below is an example of this.

```

\begin{problem}[5]
Answer this if you can!
\begin{answers}{2}
\bChoices
    \Ans0\label{testsqFirst} This is a possible answer.\eAns
    \Ans1\label{testsqSecond} Try this one (the correct one).\eAns
    \Ans0 This is an answer.\eAns
    \Ans0 Another alternative.\eAns
\end{Choices}
\end{answers}
\begin{solution}
We reference alternatives (\REF*{testsqFirst}), an incorrect answer,
and (\REF{testsqSecond}), the correct answer.
\end{solution}
\end{solution}
\end{problem}

```

Notice that the gray letters was not turned to off until after the usage of `\REF`.

Important The gray letters are typeset into the document. Do not use a background color for the checkboxes as this color will cover up the gray letters. The default background color checkboxes is transparent, keep it that way.

When typesetting an exam for paper (using the `forpaper` option), the gray letters appear as black letters. If you want actual gray letters, you have to use the `forcolorpaper`

option. In this case, you'll see the blue color appearing in various places. Change these blue colors to black using the following commands in the preamble:

```
\proofingsymbolColor{black}
\instructionsColor{black}
\eqCommentsColor{black}
\universityColor{black}
\subjectColor{black}
\linkcolor{black}
\nolinkcolor{black}
```

7.10. Displaying Points between two Markers

Some instructors might like to subdivide the exam into segments (or parts) and to have a total for that segment of problems. The `eqexam` package offers three commands for that purpose.

```
\placeMarkerHere{<name>}
\calcFromMarkers[<formatting>]{<name2>}{<name1>}
\markerTotalFmt{<formatting>}
```

Place `\placeMarkerHere` outside of a `problem`/`problem*` environment, giving each a unique name; for example `\placeMarkerHere{priorQuadForm}`. Place `\calcFromMarkers` where ever you wish a calculation to be displayed, for example,

```
\placeMarkerHere{priorQuadForm}
\begin{eqComments}[Quadratic Formula\calcFromMarkers{priorGraphing}{priorQuadForm}.]
Solve each of the following equations using the quadratic formula.
\end{eqComments}
...
\begin{problem}[5]
...
\end{problem}
...
% Finished with problems that use the quadratic formula, now create another marker
% for the next set of questions.
\placeMarkerHere{priorGraphing}
...
```

After you \LaTeX three times (and the totals are all brought up to date, the header of the **eqComments** should read **Quadratic Formula (12 points)**, there the **12 points** are the total of all points assigned between the `priorQuadForm` marker and the `priorGraphing` marker.

The formatting for the total points between markers is determined by the optional first parameter of `\calcFromMarkers`, and if there is no optional first parameter, by a global command, `\markerTotalFmt`, which sets the default formatting. The default definition of `\markerTotalFmt` is

```
\markerTotalFmt{ (\themarkercnt\space points)}
```

The command `\themarkercnt` references the counter `markerCnt` in which the calculations are made. Any redefinition of `\markerTotalFmt` should use `\themarkercnt` to reference to value.

You use the optional first parameter the same way as the definition of `\markerTotalFmt`. You can say, for example, you can type

```
\calcFromMarkers[ $\themarkercnt^{\text{pts}}$]{priorGraphing}{priorQuadForm}
```

to get a formatted total 12^{pts} , which typesets to ‘12^{pts}’.

You might have noticed that I’ve inserted a space character at the beginning of the definition `\markerTotalFmt{ (\themarkercnt\space points)}`, and place `\calcFromMarkers` up against the previous word, as in

```
Quadratic Formula\calcFromMarkers{priorGraphing}{priorQuadForm}.
```

This is so that when the required totals are not defined—early in the \LaTeX process—there is no space between `Formula` and the period (.); this is nothing but a cosmetic trivial point. After you \LaTeX enough times, the full expansion appears as,

```
Quadratic Formula (12 points).
```

7.11. Extending the `\fillin` Command

When the document author uses the `usexkv` option, and the `xkeyval` package is found on the document author’s system, the `\fillin` command is redefined to use key-value pairs in the optional first argument. The syntax for `\fillin` now is,

```
\fillin[
  underline=true|false,u,b,boxed=true|false,boxpretext=<text>,
  align=l|r|c,boxsize=\tiny|..\normalsize|\large|...\Huge,
  color=<namedcolor>,format=<\bfseries|\ttfamily|\Large|whatever>
]{<width>}{<answer>}
```

The first optional parameter uses a key-value system. The keys are described below. The second parameter `<width>` is the amount of horizontal space to leave for the student to write in the response. The third argument, `<answer>`, is the correct answer; this correct answer appears when the document is compiled with the `answerkey` option.

The description of the key-value pairs for `\fillin`:

underline: A Boolean switch, which if `true`, the fill-in region is underlined. The default is `false`, the region is not underlined.

u,b: Legacy options. If `u` is chosen, the region is underlined, if `b` is chosen, the region is not underlined. Use of the `underline` key is recommended.

boxed: A Boolean switch, which if `true`, the fill-in region is boxed in using the `\boxed` command of `amsmath` package. The default is `false`, the region is not boxed.

boxpretext: A key that takes `<text>` as its value. This value will be placed in front of the third argument, labeled `<answer>` above. The `<text>` appears in the box even when the `answerkey` is not in effect. This option is ignored if the `boxed` option is not taken.

This option allows you to create an expression like

$$y =$$

and with `answerkey`

$$y = 2x^2 + 1$$

boxsize: This is a choice key, the choices being `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge` and `Huge`. the smaller sizes probably are not useful, I give them to you for free. This key allows you do adjust the height of the box. This key is ignored if the `boxed` key does not appear in the option list. For example,

`boxsize=Large:`



`boxsize=Huge:`



Choice of size depends on the height of the anticipated response of the student. The default is `normalsize`.

align: A key that takes one of three values, `l`, `r`, and `c`. This key aligns the text within the fill-in field (when the `answerkey` option is taken): `l` (left-aligned), `c` (center, the default), `r` (right-aligned). This parameter affects the position of the `<answer>`, and does not affect the position of the `<text>`, which is aligned left, of the `boxpretext` key. The alignment becomes visible when the `answerkey` option is in effect.

color: The value of this key is a named color. The `<answer>` appears in this color, when `answerkey` option is in effect. The default is red.

format: A general purpose formatting key. The value of the key can be most anything: `\bfseries` (to change font series), `\ttfamily` (to change font family), `\Large` (to change size of the `<answer>` and `<text>` (the value of `boxpretext`)). Several formatting commands can appear as the value; thus, `format={\bfseries\Large}` makes the answer, when `answerkey` is in effect, appear in large bold font. The default is `format={\bfseries}`.

The keys are processed by an `\edef`, this allows you to define a command with your favorite options; for example, you can define

```
\newcommand{\myBoxOpts}{boxed,boxsize=Large,align=l}
```

then in the exam, type

```
$$\frac{\sqrt{-18}}{\sqrt{6}}=\fillin[\myBoxOpts]{1.5in}{3\imath}$$
```

When the `boxed` key is used, the `\boxed` command of `amsmath` is used. This command needs to be in math mode. If the `boxed` key is used, `\fillin` puts the `<answer>` in math mode. Thus, in the example above, the `<answer>` argument (`3\imath`) does not need to be inserted in math mode. If you want text to appear in the boxed in area, you can use the `\text` command

```
The first president of the US is
\fillin[boxed,align=c]{2in}{\text{Washington}}
```

Note that the `boxed` key is specified, but `\fillin` has not been placed in math mode. The `\fillin` code smart enough to automatically puts `\fillin` in math mode (using `\ensuremath`). The `<answer>` argument needs to be put into text mode. Boxing this answer does not seem to be a good choice, perhaps underlined would be better:

The first president of the US is
`\fillin[underline,align=c]{2in}{Washington}`

Now we don't need `\text` for the `<answer>` because the `boxed` key is not specified, and so `\fillin` will be typeset the `<answer>` in "text-mode."

On the other hand, if the `boxed` key is *not* used, the `<answer>` is typeset in "text-mode," unless you want it placed in math mode, in which case you need to insert `$`'s around the answer, like so

The area of a circle of radius 2 is `\fillin{.5in}{4π}`