THE UNIVERSITY OF AKRON
Theoretical and Applied Mathematics

The AcroTEX eDucation Bundle

D. P. Story

Directory

Last Revision Date: March 13, 2002 Version 4.0

Table of Contents

Preface

1. Introduction

The AcroTEX eDucation Bundle, read "AcroTEX Education Bundle", is a collection of LaTeX macro files, along with various support files and sample files. The overall theme of this bundle is ePublication in the education sector using LaTeX as the authoring application and Adobe's Portable Document Format (PDF) as the file format of the output document.

Currently, there are three components to the bundle, with others planned:

1. The web package is used to create an attractive, easy-on-the-eye page layout suitable for the WWW (or classroom/conference presentations).
2. The exerquiz package makes it very easy to create interactive exercises and quizzes.
3. The insdljs package allows for the automatics insertion of document level JavaScript. Document authors can use insdljs to cus-

tomize the processing of the exerquiz quizzes. See the documentation that accompanies the package (`insdljs.dtx`) and see also the sample file `jqzspec.tex` for an extensive example of how to modify the exerquiz macros.

4. The dljslib package is used as a library of JavaScript functions. Some type question require special processing. A JavaScript function written to process a particular function can be stored in the library, then retrieved as needed. See the documentation contained in the file `dljslib.dtx`, and try the test file for this package, `jslib_ex.tex`.

The AcroTEX Bundle should be useful to educators who want to post interactive materials for their students on the WWW.

Here is an important point that should be emphasized early in this manual. AcroTEX only supports three ways of producing a PDF document: (1) the Adobe Acrobat Distiller (version 4.0 or higher, version 5.0 or higher strongly preferred); (2) pdftex; (3) dvipdfm. In the case of (1), you probably use `dvips` to create a postscript file before distilling. Some users have tried to use GhostScript to produce

a pdf document from AcroTEX; this will not work! (You will get the PDF document but not much functionality.)

Please contact me at dpstory@uakron.edu should you encounter any problems, or have suggestions to make.

▶ See 'Getting Started' on page 18 for instructions on how to get up and running.

1.1. A Brief History

The web and exerquiz packages were written in preparation for a two-day workshop on LaTeX/PDF that I gave at the College of the Redwoods, April 30-May 1, 1999, at the invitation of David Arnold. The workshop forced me to take many of the basic macros that I had developed in plain TeX and convert them to LaTeX.

Significant additions to the exerquiz were made immediately following the 20th Annual Conference of the TeX User's Group (TUG), in August, 1999, Vancouver, British Columbia, which I attended.

The insDLJS package was written for the 22nd Annual Conference of the TeX User's Group (TUG), in August 2001, The University of

Delaware, Newark, Delaware.

1.2. Thanks

Noel Vaillant, www.probability.net, deserves my thanks for his enthusiasm for the web style file and his initial work on it inspired me to make a serious effort at writing a LaTeX package.

Thanks also goes out to Jean-Michel Sarlat for writing a French version of the web and exerquiz packages, see his Syracuse Web site. He urged me to include a language option. Thanks also goes to Michael Wiedmann who suggested a language option many months earlier, but I'm afraid, it landed on deaf ears at the time. These two provided the translations for the `french` and `german` options. (January 1, 2000)

My thanks to Heiko Oberdiek, who took a close look at insdljs. He made several suggestions, and urged me to make some significant improvements to this package.

1.3. What's New

The following is a brief enumeration of some of the major new features of the web and exerquiz packages.

- **Web**
  1. Added a `textures` option, due to Ross Moore.
  2. Made some minor changes to make web work better with the `book` document class.
  3. The introduction of the polish option. This makes a total of nine localizations of web/exerquiz: french, german, norsk, dutch, spanish, italian, russian, dansk and polish.
  4. Set an alias between `\marginsize` and `\margins`. The latter is used by pdfscreen. This will make it easier for people to switch between using web and pdfscreen. The use of `\marginsize` is now deprecated.

- **Exerquiz**

See the file eqchange.txt for more details on the change history.

1. Eliminated the need for the `\ifforAcroV` switch in the `*.ins` files. I've written new code that detects whether Acro4 or Acro5 is being used; in the latter case, a `try/catch` strategy is used to catch exceptions.
2. Added a `textures` option, due to Ross Moore.
3. Wrote same JavaScripts and placed them in the `dljslib` to process questions in which an equation is the expected answer. See the sample file `jqzequat.tex`.
4. Wrote the `dljslib` Package, this is a library of JavaScript functions that extends the capabilities of exerquiz.
5. Extended math fill-in questions to include arbitrary multivariate questions. See the sample file `multivar.tex`.
6. Added several macros: `\defaultquiztype` and `\quiztype`. The first command takes no arguments, the second on takes one: either l or f; e.g. `quiztype{f}` (`quiztype{l}`) causes the quiz environment to ignore the optional 'quiztype' parameter (`*` or no `*`) and to use a form-type (resp. link-type). Placing `\defaultquiztype`

reverts the quiz environment back to its default behavior (obeying the first optional parameter).

7. Added the convenience commands: `\useBeginQuizButton`, `\useEndQuizButton`, `\useBeginQuizLink` and `\useEndQuizLink`. See 'The BeginQuiz and EndQuiz Form Buttons' on page 106 for details.

8. Added the `noHiddensolutions` for `exerquiz`, and added an 'H' option for exercises to hide solutions, see Hiding some Solutions and The `noHiddensolutions` Option, for details.

9. Added a method of assigning points to a quiz question. Useful for submitting questions to a CGI for recording in a database.

10. Added a `noPeeking()` JS function and supporting macros. When the `\NoPeeking` command is executed, the student cannot see the solutions to the quizzes (not shortquizzes) by browsing through the file. See '`\NoPeeking`' on page 161.

11. Added new JS functions `lowThreshold()` and `highThreshold()`. I modified the end of quiz macros to incorporate the calling of one

of these two (or calling a document author defined routine). See 'Setting the Threshold' on page 108.

12. Added a '`debug`' option, which gets passed on to `insdljs` package. This can be used to write some debugging commands within your JS. ('The `debug` Option' on page 56)

13. For math fill-ins, the author does not use the JavaScript syntax; the author can use simplified notation, e.g, `2*x*e^(x^2)` instead of `2*x*exp(pow(x,2))`. Author's answer now passes through the ParseInput JS routine.

14. Now the document author can define a custom JS function to process an answer. Also, when writing Math fill-in questions, you can also define your own variable (no longer restricted to just 'x').

15. Added in a `solutions` environment to `quiz` environment. Added additional optional parameter into the `\RespBox`, `\RespBoxTxt` macros to indicate the presence of a solution. The solution is viewed by shift-clicking on the "Ans" Button (`\CorrAnsButton`). A button or checkbox that has a solution has a boundary color of `\solutionColor`.

16. `Exerquiz` now uses the package `insDLSJ` to insert document-level JavaScripts; this gives the document author a chance to write custom JavaScripts. I've also modified many macros that enable the document author to "hook" into.

17. New command `\RespBoxEssay` that can be used to pose Essay-type questions. The question is not evaluated by JavaScript within the document; rather, this question should be submitted to a CGI for later review by the instructor.

18. `\RespBoxMath` has been defined to be the same as `\RespBox` to give a little more consistency in naming.

19. The use of `\RespBoxNT` has been deprecated. Added in feature to customize the comparison of two answers.

20. Added a text fill-in question type that can be used in the `shortquiz` and `quiz` environments. ('The Text Question' on page 125)

21. A `preview` option has been added. When this option is used, the bounding rectangles of all form fields are framed so their positions can be seen in a dvi previewer. See 'The `preview` Option'

on page 54.

- **insDLJS**

The `insDLJS` Package is a general purpose LaTeX package for inserting Acrobat JavaScript into the document-level section of a PDF document. The package features the `insDLJS` environment. This environment typically goes in the preamble of a LaTeX source file, or in the style files of a LaTeX package. See the documentation contained within the `insdljs.dtx` file for additional details. There is a sample file, `insdljs_ex.tex` that can be used as a startup test file.

- **dljslib**

The `dljslib` Package acts as a library of JavaScript functions. Due to the increased programmability of `exerquiz` and its new found flexibility, it is possible to write a number of different routines to handle various kinds of math fill-in questions. These JavaScript functions can be stored in the library and retrieved when needed. This package requires the `insdljs` package.

See the documentation contained within the `dljslib.dtx` document, and see 'Using the `dljslib` Package' on page 146.

Now, I really must get back to work. 𝔇𝔖

2. Getting Started

There has been a new package added to the AcroTEX Bundle, the `insDLJS` Package. This package allows the document or package author to write JavaScripts to the document level JavaScript section of a PDF document. `Exerquiz` now uses `insDLJS` to place its JavaScripts into the PDF document.

▶ The program files for AcroTEX Bundle consist of `web.sty`, `exerquiz.dtx`, `exerquiz.ins`, `insdljs.dtx`, `insdljs.ins`, `dljslib.dtx`, `dljslib.ins`, and `acrotex.ins`

1. Place all these files in the same directory. This directory must be in the search path of your LaTeX system, perhaps in a separate folder called `acrotex`.
2. The whole bundle can be unpacked by latexing `acrotex.ins`.

(The other `*.ins` files are the installation files for the individual packages, `acrotex.ins` is the combined installation file.) Important: See the next section, Unpacking the AcroTEX Bundle for important information on unpacking the bundle.

3. Place the sample files either in the same folder as the AcroTEX program files, or in another folder of your choosing. See the section titled 'Sample Files' on page 20 for more details on these.

After reading the manual you are then ready to write your own set of tutorials, exams, exercises and quizzes. 𝔇𝔖

2.1. Unpacking the AcroTEX Bundle

To install the AcroTEX Bundle, you must first "unpack" it. Unpacking is performed by "LaTeXing" the file `acrotex.ins`. Simply execute `latex acrotex.ins` from the command line (the command line may vary depending on your TEX System), or if you use a TEX/LaTeX friendly editor, open the file in the editor and `latex` it.

▶ The language localizations have been commented out. Just uncomment the language you intend to use.

- Also in the `exerquiz.ins` file is the line

```
% \file{template.def}{\from{exerquiz.dtx}{copyright,template}}
```

Uncomment this line to get the template file, used for developing language localizations.

2.2. Sample Files

The following sample files accompany the distribution:

1. webeqtst.tex demonstrates the capabilities of the `exercise` environment. Examples are also given of multiple choice questions within the `shortquiz` and `quiz` environments.
2. jquiztst.tex shows off math fill-in questions for the `shortquiz` and `quiz` environments.
3. jtxttst.tex features text fill-in questions in the `shortquiz` and `quiz` environments. The file also discusses the various parameters of the `\RespBoxTxt` command.
4. jqzspec.tex is a tutorial on how to modify the way the math fill-in command `\RespBoxMath` processes the user input. This file has an

extensive example that shows how you can process, for example, a vector answer.

5. quizpts.tex is a file that shows how to assign points to questions in a `quiz` environment.
6. exlist.tex shows how to create an environment in which the exercises of an `exercise` environment are listed in enumerated form.
7. book01.tex shows the basic web layout with the `book` document class.
8. insdljs_ex.tex This is a test file of the insdljs package.
9. dljslib_ex.tex This is a test file of the dljslib package.
10. multivar.tex This is a sample file illustrating multivariate processing.
11. jqzequat.tex A sample file illustrating how equations can be processed.

Both the `shortquiz` and `quiz` environments use JavaScript to evaluate the questions. This JavaScript resides at the document-level and, with one exception, is inserted automatically into the PDF file. That

one exception is the case of using Acrobat Distiller 4.05 or less to create your PDF file from PostScript.

In the description of LaTeXing the sample files, the techniques of handling the exceptional case will be explained.

2.3. LaTeXing Your First File

The functionality of the `shortquiz` and `quiz` environments depends on JavaScript code that is placed at the "document-level", to use Adobe's terminology. The applications pdftex and dvipdfm offer direct support for writing to this document-level. For those who use the Adobe Distiller, things aren't quite so easy.

In this section, we describe how to insert document level JavaScripts into a PDF document, prepared from a LaTeX source that uses the `exerquiz` package. Even though the handling and insertion of document-level JavaScript is done with the package insdljs, a little care must be taken—at least in the Distiller case—when building your .PDF document.

Open `webeqtst.tex` in your favorite text editor. The top lines

read:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[tight,designi]{web}
\usepackage{exerquiz}
```

- **For pdftex and dvipdfm Users**

Edit the third line by inserting your driver; the choices are `pdftex` and `dvipdfm`. For example, if you use dvipdfm, the lines should read:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvipdfm,tight,designi]{web}
\usepackage{exerquiz}
```

For pdftex, you simply call `pdflatex`, and you have your nice PDF document, ready for review. The insertion of the document level JavaScript is automatic.

For dvipdfm, you LaTeX the document, then hit it with dvipdfm, and your ready to review your PDF document.

- **For Distiller Users**

Edit the third line by inserting your driver; the choices are `dvips` and `dvipsone`. For example, if you use dvips, the lines should read:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvips,tight,designi]{web}
\usepackage{exerquiz}
```

▶ **For Distiller 5.0$^+$ Users.** When you LaTeX the source file you create a `.dvi` file, and one or more `.fdf` files. The `.fdf` files (e.g., `exerquiz.fdf`) contain the document level JavaScript that needs to be imported into your document.

You then convert your `.dvi` to `.ps` using either dvips or dvipsone, and distill. Important: When you distill, save the `.pdf` back to the same folder in which your source file (`.tex`) resides as this is were the `.fdf` files reside too. Insertion of document level JavaScripts automatically takes place when you open your newly distilled document in the Acrobat application. (It is actually Acrobat that imports the scripts, not the Distiller.)

☞ When your document is opened in Acrobat for the first time, the JavaScript contained in the `.fdf` files (e.g., `exerquiz.fdf`) is imported into the document and is stored at the document level. Important: *Save your document.* When you save, the JavaScripts you just imported are also saved with the document. At this point you can move your PDF to another folder, or to the web. The document does not need the `.fdf` files any more.

▶ **For Distiller 4.0–4.05 Users.** Versions prior to 5.0 of the Acrobat product cannot import document level JavaScript contained in a `.fdf` file. The JavaScript needs to be inserted "by hand".

The procedure is as follows: Modify the preamble and inset the `acrobativ` option for the exerquiz package:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvips,tight,designi]{web}
\usepackage[acrobativ]{exerquiz}      %<- acrobativ option
```

This suppresses all the JavaScript code generation that is used in the case of pdftex, dvipdfm or Distiller 5.0$^+$. After that change, LaTeX the document, and convert the `.dvi` file to PostScript (using `dvips` or

`dvipsone`), and distill. Now, open the new PDF document in Acrobat (formerly known as Exchange). Click on `Document > Insert Pages`, browse, and choose the PDF file eq_dljs.pdf, which comes with the AcroTEX Bundle. The file is now inserted. Next, click on `Document > Delete Pages` and delete the page you just inserted! Important: Do a "`Save As`"; now you are done!

The PDF file eq_dljs.pdf contains all the standard JavaScript that goes in at the document level. In all the other situations discussed above, you can modify the JavaScript from the preamble, for example, in the exerquiz source code we have

```
\newcommand\checkColor{["RGB", 0, .6, 0]}
```

This command is expanded when the document level JavaScript is imported into the PDF document. If `\checkColor` has been redefined

```
\renewcommand\checkColor{["RGB", 1, 0, 0]}
```

it is this definition that is used in the expansion. This convenience is lost for $4.0 \leq$ distiller ≤ 4.05. You can edit eq_dljs.pdf and make whatever changes you please to the script, in terms of changing color. However, it is strongly recommended that you upgrade to version 5.0!

The Web Package

3. The Web Package

The purpose of the web package is to create a page layout for documents meant for screen presentation, whether over the WWW or classroom/conference presentations, in PDF. Such documents are *not* (necessarily) *intended to be printed*; consequently, the page layout is, in some sense, optimized for screen viewing.

3.1. Overview

The web package redefines `\maketitle` and `\tableofcontents` in a more web friendly way; it colors the section headings, and inserts `\bullets` (•) at the `\subsubsection` level. This, to my eyes, is very attractive. Additionally, certain navigation devices—a navigation bar and some direction icons—are included in the package.

There are options for a small collection of drivers: `dvipsone`, `dvips` and `pdftex`. The language option redefines certain language dependent elements of the package to other languages. Currently, the follow-

ing options are supported: `dutch`, `french`, `german`, `italian`, `norsk`, `russian spanish`, `dutch` and `polish`. There is even an option for reformatting the `web` style to a print format!

The capabilities of the web package and its options are discussed below. Any comments and suggested improvements (new features) would be greatly appreciated.

3.2. Package Requirements

The web package was designed for screen presentations tutorials, such as classroom or conference lectures, short technical articles, etc.; consequently, the `article` class of LaTeX seems to be a sufficient for these purposes. Though you can use web with any of the standard classes that define the `\section`, `\subsection` and `\subsubsection` commands, the package is really meant to be used with the `article` class. It is **strongly** suggested!

The package heavily depends on Sebastian Rahtz' hyperref package (now maintained and developed by Heiko Oberdiek). The web package was developed using version 6.56 of hyperref. Using prior versions of

hyperref *may* lead to successful compilation—no guarantees offered. It is best to work with the most recent version of hyperref.

The color and amssymb packages are also required. The former is for obvious reasons, the later is to provide certain navigational symbols when the `navibar` option is invoked.

Finally, to create quality PDF document, type 1 fonts *must* be used. Fortunately, type 1 fonts in the Computer Modern font set are freely available, and come with all the major freeware, shareware and commercial TEX systems. If you haven't done so already, learn how to use the type 1 fonts.

In this regard, I have written an article that may be of interest to you entitled "*Using LaTeX to Create Quality PDF Documents for the WWW*", see reference [10].

3.3. Basic Usage

To use the web package, insert into the preamble of your document the following:

```
\usepackage[<driver_option>,<other_options>]{web}
```

Replace `<other_options>` with any of the options recognized by web; see Section 10 for a complete list of options. The optional argument `<driver_option>` above defines the driver to be used; for example,

```
\usepackage[dvipsone]{web}
```

Currently, the web package supports six drivers: `dvipsone`, the dvi-to-ps converter by Y&Y, Inc., (http://www.yandy.com/); `dviwindo`, Y&Y's `dvi`-previewer; `dvips`, the freeware dvi-to-ps converter; `pdftex`, the `tex`-to-`pdf` application; and `dvipdfm`, the `dvi`-to-`pdf` application by Mark Wicks, (http://odo.kettering.edu/dvipdfm/); and the commercial TEX system for the Mac, `textures`.

▶ The package has been tested using `\documentclass{article}` and it is *strongly* recommended that this class be used.

- **Setting the Driver Option**

You can set your driver option in one of three ways:

- Pass as a local option:
  `\usepackage[<driver_option>]{web}`

- Pass as a global option:
  `\documentclass[<driver_option>]{article}`
- Create the file `web.cfg` with the single command in it:
  `\ExecuteOptions{<driver_option>}`
  Place the file `web.cfg` in any folder where LaTeX looks for input files. Then, you need only type `\usepackage{web}`.

Where `<driver_option>` is any of the following options: `dvipsone`, `dviwindo`, `dvips`, `pdftex`, `dvipdfm` or `textures`

The macros of the web package have been extensively tested using the Y&Y TeX System (www.yandy.com) for the `dvipsone` and `dviwindo` options and a MikTeX System (www.miktex.org) for the `dvips`, `pdftex` and `dvipdfm` options.

• The `tight` Option

In an effort to compact more material per page, I've introduced a `tight` option. When this option is used, many of the list parameters are redefined so that is not so much space around these environments, and between items.

```
\usepackage[<driver_option>,tight,<other_options>]
```

This screen version of this manual was typeset with the `tight` option, the print version was typeset without it.

3.4. Setting Screen Size

Beginning with version 2.0, the screen size can be set by the author. There are two ways to do this: (1) use the macros `\screensize` and `\margins` (These are the same macros—slightly redefined—for setting the screen size used by Radhakrishnan in his fine sceen package pdfscreen.); use a screen design option. The next two sections addresses each of these in turn.

- **Custom Design**

There are five dimensions that need to be specified. As with pdfscreen, the two commands `\screensize` and `\margins` are used for doing so.

The command `\screensize` takes two length parameters:

```
\screensize{<height>}{<width>}
```

The <width> and <height> parameters are desired screen size of the page. The screen version of this manual uses

```
\screensize{3.72in}{4.67in}
```

The other command, \margins, which determines the desired margins, takes four length parameters:

```
\margins{<left>}{<right>}{<top>}{<bottom>}
```

The values of \textheight and \textwidth are computed based on the screen size and the margins. The margin settings for this document are given below:

```
\margins{.25in}{.25in}{30pt}{.25in}
```

▶ An important comment about the third parameter <top>. As with pdfscreen, we put \@Topmargin=<top>. The running header fits within the top margin (this varies from standard LaTeX practice). The web package dimension \web@Topmargin is the distance from the top of the screen down to the top of the running. Thus,

```
\@Topmargin = \web@Topmargin + \headheight + \headsep
```

Also, `\web@Topmargin` can be used to adjust the positioning of running header, which is specified in the `\margins` command. The default value of `\headheight` is `8pt`, so the value of `\headsep` is determined by the above equation. See the web.sty file for more details.

- **Screen Design Options**

You your convenience, I've included three options, `designi`, `designii` and (you guessed it) `designiii`. The first one roughly corresponds to the original screen dimensions of web. The other two set the screen dimensions at 4.5in×5in and 5in×6in (height × width), respectively. You can type

```
\usepackage[designi,pdftex]{web}
```

to obtain the standard web dimensions.

▶ When you specify a screen design, the macros `\screensize` and `\margins` are redefined to gobble up their parameters. To define a custom screen size, therefore, do not specify a screen design option for web.

3.5. Hyperref Options

The web package loads hyperref into the document and sets some selected options of that package; therefore, including the hyperref package is not needed in the preamble of your own document.

Any additional hyperref options that are needed can be introduced into the package using hyperref's `\hypersetup` macro, for example,

```
\documentclass{article}
\usepackage[dvipsone]{web}   % or dvips or pdftex

% Declare additional hyperref options using \hypersetup
\hypersetup{pdfpagemode=None,bookmarksopen=false}
```

Documentation of the options that hyperref recognizes can be had by either LaTeXing the file `hyperref.dtx`, or by getting a copy of the *The LaTeX Web Companion* [5] by Michel Goossens *et al.*

3.6. The Title Page and TOC

The title page is constructed from the values of the macros: `\title`, `\author`, `\university`, `\email`, and `\version`. The values of some

of the macros `\title` and `\author` are also transferred to the PDF-DocInfo section of the Acrobat Reader/Exchange.

Additionally, the values of `\subject` and `\keywords` are inserted into the PDFDocInfo section.

- **Basic Information Macros**

Just fill in the values of all the basic macros briefly described above. For example, the following is a copy of the title information for this document:

```
% \title,\author,\subject,\keywords are sent to DocInfo
\title{The Web and Exerquiz Packages Manual of Usage}
\author{D. P. Story}
\subject{How to create on-line exercises and quizzes}
\keywords{LaTeX,hyperref,PDF,exercises,quizzes}

% \university,\email,\version are used only on title page
\university{THE UNIVERSITY OF AKRON\\
   Mathematics and Computer Science}
\email{dpstory@uakron.edu}
\version{1.30}
\copyrightyears{1999-2002}
```

▶ The `\title`, `\author`, `\subject`, `\keywords` are a convenient way of entering information in the Document Information fields—see

`File > Document Info > General...(Ctrl+D)`

in the Acrobat Reader/Exchange.

If `\title` contains control sequences that do not expand to the Standard PDFDocEncoding character set, Distiller will be thrown into a tailspin; hyperref defines the `\texorpdfstring` macro[1] to avoid these kinds of problems. For example,

```
\title{The \texorpdfstring{$e^x$}{EXP} Function}
```

The first argument is the one that is typeset (on the title page, the title of the document will be 'The e^x Function'); the second argument is the one that is sent to the title field of DocInfo in the Acrobat Reader (and will read 'The EXP Function').

Of all the Basic Information Macros, use `\texorpdfstring` only with the `\title`, `\author`, `\subject` and `\keywords`, all of which are used in the DocInfo field of the Acrobat Reader.

[1] The code for handling PDFDocEncoding for hyperref is due to Heiko Oberdiek

▶ `\texorpdfstring` works for `\section`, `\subsection`, etc. as well.

Having entered the information you can now type the standard sort of LaTeX commands of `\maketitle` and `\tableofcontents`:

```
\begin{document}
\maketitle
\tableofcontents
...
...
\end{document}
```

▶ Use the file `webeqtst.tex`, which comes with the distribution, as a prototype or template for your own document.

• Redefining `\maketitle`

The arguments of the Basic Information Macros macros, as just discussed, are used to define text macros with no parameters; for example, when you type `\title{Web Package}`, the macro `\title` takes its argument and defines a macro `\webtitle` that expands to 'Web Package'.

You can redesign the title page to suit your needs simply by redefining the `\maketitle`: rearrange the macros listed in the second column of Table 1 on the page, or include a graphic, or change the background color. To redefine `\maketitle`, use the commands:

```
\renewcommand\maketitle{...your design...}
```

See the definition of `\maketitle` in the `web.sty` file for an example.

This macro	defines this macro
`\title`	`\webtitle`
`\author`	`\webauthor`
`\subject`	`\websubject`
`\keywords`	`\webkeywords`
`\university`	`\webuniversity`
`\email`	`\webemail`
`\version`	`\webversion`
`\copyrightyears`	`\webcopyrightyears`

Table 1: The Basic Information Macros

When making the design, it is useful to know that the web package

uses `\hypertarget` to create a named destination, '`webtoc`', in the table of contents, Use this `webtoc` to jump to the table of contents using the macro `\hyperlink`.

Lastly, I have included a macro, `\optionalpagematter`, you can use to include additional material on the title page. Here is an example of usage:

```
\renewcommand\optionalpagematter{\vfill
    \begin{center}
    \fcolorbox{blue}{webyellow}{
    \begin{minipage}{.67\linewidth}
    \noindent\textcolor{red}{\textbf{Abstract:}} This
    file attempts to teach you how to create a simple
    \LaTeX\ document.
    \end{minipage}}
    \end{center}}
```

The above definition will create the framed box seen below.

Abstract: This file attempts to teach you how to create a simple LaTeX document.

The `\optionalpagematter` appears just below the `\webauthor` and

above the directory listing. See the sample file `webeqtst.tex` for an example of this feature.

- Of course, you can rearrange everything.

- **The `nodirectory` option**

The inclusion of `\tableofcontents` is optional. The article you write may be short, or perhaps it may just be a collection of exercises and quizzes. In this case, you may not want a table of contents.

If you do not want a table of contents, you would not include `\tableofcontents` just after `\begin{document}`. Without a table of contents, you may as well turn off the directory listing on the cover page as well. Use the `nodirectory` option to do this:

```
\usepackage[dvips,nodirectory]{web} % dvipsone, pdftex
```

The directory listing does not appear on the title page.

- **The `latextoc` option**

If you don't like the default design for the table of contents, you can always recover the standard LaTeX table of contents by using the

`latextoc` option with the web package:

```
\usepackage[latextoc]{web}
```

Should you want to go with this option, you might consider including

```
\hypersetup{linktocpage}
```

Look at the table of contents with and without this hyperref option to decide which you prefer.

3.7. Navigational Aids

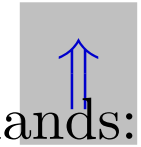
The web package offers a couple of navigational aids to help you move around: the `navibar` Option, and some direction icons.

- **A Navigational Bar**

Use the `navibar` option of web to add a navigational toolbar, as seen at the bottom of this page. Usage:

```
\usepackage[<driver_option>,navibar]{web}
```

the result is the navigation bar you see at the bottom of the page.



▶ The toolbar can be turned on or off by the following commands: `\NaviBarOn` and `\NaviBarOff`. The navigational toolbar at the bottom of the page was generated by the `\NaviBarOn`. `\NaviBarOff` was placed on the next to turn off the bar.

- **Direction Icons**

The up arrow you see in the upper right-hand corner was constructed using colored rules and the AMS symbol font, `amssymb`. The uparrow icon was produced by the command:

```
\insertnaviiconhere{\ArrowUp{\hyperlink{webtoc}}}
```

Or, more generally,

```
\insertnaviiconhere{\ArrowUp{link_command}}
\insertnaviiconhere{\ArrowDown{link_command}}
```

This will insert direction icons on the current page (I hope).

If you want a running direction icon you can use

```
\insertnaviiconhereafter{\ArrowUp{link_command}}
```

or

```
\insertnaviiconhereafter{\ArrowDown{link_command}}
```

- To discontinue a running arrow icon type

  `\defaultpageheader`

one the page you want the arrow(s) to disappear.

3.8. The Language Options

The language options redefine all of the language dependent text macros that appear on the title page, in the table of contents and in the running headers. Invoke these options in the usual way:

```
\usepackage[<driver_opt>,<lang_opt>]{web}
```

Where, `<lang_opt>` is one of the following: `dutch`, `french`, `german`, `italian`, `norsk`, `russian`, `spanish` and `polish`.

The web and exerquiz packages seem to be compatible with the babel package; you can use

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage[dvips,french]{web}
\usepackage{exerquiz}
```

subject to the usual restrictions on these language packages. (Don't use characters declared active by these languages within a `\label`, or as a field name for a quiz.

The translations for the `french` option is due to the tremendous efforts of Jean-Michel Sarlat, and Michael Wiedmann did the translations for the `german` option.

3.9. Paper Related Options

- **The `forpaper` option**

Some people may want to create exercises using the `exercise` environment for a paper document. The `forpaper` option can be used to remove the color from the document (back to black and white `:-`), and restores the standard `\textheight` of a standard `article` class LaTeX document. The `\textwidth` will be the same as determined by your `\screensize` and `\margins` parameters or your design option (see Screen Design Options) so the line breaks are the same for the "web" version and the "print" version.

Using `forpaper` with the `latexlayout` option will give you the

standard LaTeX `\textwidth`.

The `forpaper` option also changes the `\newpage` command to `\par\medskip` at the end of each solution—we don't want to waste paper now do we.

Finally, there is a boolean switch `\ifeqforpaper`, which you are free to use to refine the look your `forpaper` version.

- **The `latexlayout` option**

For those who want to go "totally native", use the `latexlayout` option with the `forpaper` option. When the `latexlayout` option is used, the page layout redefinitions of web are bypassed, leaving the original layout values of the `article` class of LaTeX.

The Exerquiz Package

4. Overview

The exerquiz package provides environments for creating the following interactive elements in a PDF document.

- The `exercise` Environment: Macros for creating on-line exercises.
- The `shortquiz` Environment: Macros for creating interactive quizzes with immediate feedback.
- `shortquiz` with Solutions: Macros for creating quizzes with immediate feedback and a link to the solutions to the quizzes.
- The `quiz` Environment: Macros for creating quizzes graded by JavaScript, with an option to have the quizzes corrected using JavaScript.

In each of the quiz environments, you can pose multiple choice, math fill-in, or text fill-in questions.

The exerquiz provides the above listed environments for the `dvips-one`, `dvips`, `textures`, `pdftex` and `dvipdfm` options. For the case of

the `dviwindo` option, only the `exercise` environment is available.

There are options for reformatting the exercises to a print format; for excluding the solutions to the exercises; for writing the solutions to the exercises so they follow the question; for different languages, and much more.

The exerquiz also allows you to rearrange the order and location of the solutions to the exercises and quizzes; to redefine many running headers; to customize the exercises and quizzes; and to use the `exercise` environment to create a new environment with its own counter—or with no counter at all.

All the above mentioned macros and the options of the package are discussed in this section.

4.1. Exerquiz and Acrobat JavaScript

Exerquiz now uses the insDLJS Package to insert Document-level JavaScripts into the PDF file. The quizzes created using the `shortquiz` or `quiz` environment are graded, marked and scored using these inserted JavaScript functions.

Because the package insDLJS is already loaded, it is easy for the document author to develop JavaScripts that can be called from the standard Exerquiz commands. The ability to write JavaScript, therefore, right in the LaTeX document gives a unique programming flair to Exerquiz.

4.2. Package Requirements

The exerquiz package is independent of the web package; however, exerquiz utilizes hyperref just as web does. Use the latest version of hyperref. In addition to the color package, also used by web, exerquiz also uses the sverbatim package, this is used to write verbatim solutions to exercises and quizzes to certain auxiliary files.

Results from the quizzes created by the `shortquiz` and `quiz` environments are evaluate using Document-level JavaScripts. These JavaScripts are inserted into the final PDF file using the insdljs package. This package makes it easy for the package writer or document author to write JavaScripts.

The exerquiz package uses *form features* of PDF that web does not

use. For the interactive features to properly work, use Acrobat Reader 4.0 or higher.

4.3. Basic Usage

Place in the preamble of your document

```
\usepackage{exerquiz}
```

- Use exerquiz with the web package:

```
\usepackage[<driver_option>,<more_options>]{web}
\usepackage[<options>]{exerquiz}
```

A complete list of the options recognized by exerquiz can be found in Section 10; they are also discussed below.

No driver option with exerquiz is needed if you are using the web package. (The driver options for the web package are `dvipsone`, `dvips`, `pdftex`, `dvipdfm`, `dviwindo` and `textures`.)

For the `dvipdfm` option to work properly you will need dvipdfm, version 0.12.7b or later, and hyperref, version 6.68a or later.

- Use hyperref and exerquiz with either `dvipsone` or `dvips`:

```
\usepackage[<driver_options>,<more_options>]{hyperref}
\usepackage{exerquiz}
```

Permissible driver options are `dvipsone` and `dvips`.

- Use hyperref and exerquiz with `pdftex`, `dviwindo` or `dvipdfm`

```
\usepackage[<driver_options>,<more_options>]{hyperref}
\usepackage[<driver_option>]{exerquiz}
```

See the next few paragraphs for more details.

- **The `pdftex` Option**

The exerquiz package is independent of the web package. Therefore, you can create your own page layout package and use exerquiz to help you create exercises and quizzes. Of course, hyperref must be used.

Should you want to use the exerquiz package using pdftex without the web package, use the `pdftex` option:

```
\usepackage[pdftex,<more options>]{hyperref}
\usepackage[pdftex]{exerquiz}
```

In particular, pdfscreen[2], a screen design package written for pdftex

[2] CTAN:macros/latex/contrib/supported/pdfscreen

by C. V. Radhakrishnan, has been tested and works correctly with exerquiz. For example,

```
\usepackage[screen,article,sidebar]{pdfscreen}
\usepackage[pdftex]{exerquiz}
```

See the sample file `eq_pdfs.tex` already set up for use with pdfscreen, obtained by downloading the zipped file `eq_pdfs.zip`.

- **The `dvipdfm` Option**

Should you want to use the exerquiz package without the web package, in this case, the usage is

```
\usepackage[dvipdfm,<more_options>]{hyperref}
\usepackage[dvipdfm]{exerquiz}
```

- **The `dviwindo` Option**

Beginning with version 1.3 of web and exerquiz, dviwindo (the `.dvi` previewer by Y&Y, Inc.) is supported. This means that hypertext links will be active from within the dviwindo previewer—as will as from within the Acrobat Reader after the file has been converted to PDF.

Should you want to use the exerquiz package without the web package, in this case, the usage is

```
\usepackage[dviwindo,<more_options>]{hyperref}
\usepackage[dviwindo]{exerquiz}
```

▶ **Important Note:** *Only* the `exercise` environment (the material described in Section 5) is supported by these two options. None of the quiz environment can be used with these two options at this time. Y&Y users need to use the `dvipsone` option if the a quiz environment is needed.

- **The Language Option**

The language option, available in the web package, can be invoked even when the web package is not used.[3] Currently, `dutch`, `french`, `german`, `italian`, `norsk`, `russian`, `spanish` and `polish` are the supported options. For example, with hyperref, you could use:

```
\usepackage[<driver_option>,<more_options>]{hyperref}
\usepackage[<driver_option>,french]{exerquiz}
```

[3] Otherwise, the language option is introduced as an option of the web package.

Where <driver_option> is any of the following drivers: dvipsone, dvips, pdftex, dviwindo or dvipdfm. *Note*: the <driver_option> is not needed with the exerquiz package with dvipsone or dvips.

- **The forpaper Option**

The forpaper option, also available in the web package, is needed in the exerquiz package if your are using exerquiz without web. The option is invoked in the usual way

```
\usepackage[<options>]{hyperref}  % or pdfscreen
\usepackage[forpaper]{exerquiz}
```

See the discussion of the forpaper on page 45 given earlier.

- **The preview Option**

The exerquiz package can generate a large number of form fields: buttons, check boxes, radio buttons and text fields. These are PDF objects and cannot be seen in a dvi previewer. By using the preview option, the bounding rectangles of the form objects are surrounded with rules, which outlines the form fields and makes their positions visible.

This option may help you to fine tune the positions of the form fields. The option is for developmental use only. When you are satisfied with the positioning and are ready to publish, remove this option.

▶ This option is not useful with the `pdftex` option, as pdftex does not (normally) produce a dvi file.

- **The `nodljs` Option**

If you are creating a document that is meant to be printed or your document only has exercises and solutions in it (which do not require JavaScript), the size of the document can be reduced significantly by using the `nodljs` option. This option is just passed on to the insdljs package.

- **The `acrobativ` Option**

If the document author is using the `dvips` or the `dvipsone` option but has only Acrobat 4.0 or 4.05, then the document level JavaScripts need to be inserted manually. Therefore, we need to turn off the automatic inclusion of JavaScript. This option does exactly that; it is equivalent to the `nodljs` option.

- **The `exercisesonly` Option**

If the document author only uses the `exercise` environment, then all the document-level JavaScripts of `exerquiz` are not needed. Use either one of these two equivalent options to exclude the insertion of the JavaScripts.

This is a convenience option that simply call the `nodljs` option described above.

- **The `debug` Option**

Developing JavaScript functions can be tricky. Quite often, it is useful to insert some code lines that will help you in debugging a particular function or set of functions. For example, you might want to verify that the parameters being passed to a function are the correct ones, or that the return value is correct. You can have Acrobat write the values to its console like so:

```
console.println("Function myFunc");
console.println("Parameters: x = " x + ", y = " + y );
console.println("Return Value: retnValue = " + retnValue);
```

In the above code, I have used the `console.println()` method, which is only available in the Acrobat application, not the Reader. For the Reader, one could use `app.alert()`, but this method is not well-suited for monitoring values of a large number variables as the script executes. If you don't have the full Acrobat, the `debug` option will not be useful.

Exerquiz just passes this option on to the insDLJS package. Additional details on the debug option can be found there. Within the `insDLJS` environment, you can place debugging code lines as follows:

```
function myFunc(x,y)
{
    retnValue = x + y;
\db console.println("Function myFunc");\db%
\db console.println("Parameters: x = " x + ", y = " + y );\db%
\db console.println("Return Value: retnValue = " + retnValue);\db%
    return retnValue;
}
```

Any line that begins with `\db` and ends with `\db` is a debugging line. These lines will be included if the `debug` option is taken; otherwise they are removed. The '`%`', is the comment character within the

insDLJS environment, and prevents, in this case, the introduction of a carriage return.

5. The exercise Environment

The exerquiz package defines **exercise** and **solution** environments, the latter being nested inside the former. With these environments, you can create questions (exercises) with solutions. Solutions are written **sverbatim** to the auxiliary file **\jobname.sol**, then input back in near the end of the document. A hypertext link is created to connect the exercise with the solution.

An exercise with multiple parts can also be defined, with hypertext links to the solutions to the individual parts.

The **exercise** environment has its own counter (**eqexno**), but there is an option for using another counter—or no counter at all. This may be useful for creating a numbered example environment.

There is an option for placing the solutions immediately after the statement of the problem. This, again, may be useful for an example environment where you want the solution to the example to follow the

statement, rather than being hypertext-linked to the solution.

Finally, there is an option for hiding solutions, in the following sense: When the hidden option is used, the solutions are commented out rather then being written to the `\jobname.sol` file. Additionally, there is a global option, `nohiddensolutions`; in this case, when you re-LaTeX, the solutions are written to `\jobname.sol`, and input back into the document.

5.1. Basic Usage

The syntax for the `exercise` and `solution` environments is as follows:

```
\begin{exercise}
Your Question.
\begin{solution}
The Solution to Your Question
. . . . . .
. . . . . .
. . . . . .
\end{solution}
\end{exercise}
```

Here is an example of the usage.

EXERCISE 1. Evaluate the integral $\displaystyle\int x^2 e^{2x}\,dx$.

The code for this is

```
\begin{exercise}\label{ex:int}%
Evaluate the integral \(\displaystyle\int x^2 e^{2x}\,dx\).
\begin{solution}
We evaluate by \texttt{integration by parts}:
\begin{alignat*}{2}
 \int x^2 e^{2x}\,dx &
   = \frac12 x^2 e^{2x} - \int x e^{2x}\,dx &&\quad
           \text{$u=x^2$, $dv=e^{2x}\,dx$}\\&
... lines removed ...
   = \frac14(2x^2-2x+1)e^{2x} &&\quad
           \text{simplify!}
\end{alignat*}
\end{solution}
\end{exercise}
```

See the demo file webeqtst.tex for a complete listing of this exercise.

- Questions and solutions are kept together *à la Knuth*. The solu-

tions are written to the file `\jobname.sol` verbatim then input back using the macro `\includeexersolutions`.

▶ You can redefine the counter to include the section number. For example.

```
\renewcommand{\theeqexno}{\thesection.\arabic{eqexno}}
```

can be placed in the preamble of your document. In this case, the above exercise would appear as Exercise 5.1.

▶ The usual cross-referencing mechanisms for LaTeX, i.e., using `\ref` and `\pageref`, work as expected.

For example, the label '`\label{ex:int}`' was placed just after `\begin{exercise}` on the previous page, let us now reference Exercise 1, on page 60.

```
let us now reference Exercise~\ref{ex:int},
on~\pageref{ex:int}.
```

Of course, the nicer looking variations can be done as well: For example, see Exercise 1.

```
\hyperref[ex:int]{\textsc{Exercise~\ref*{ex:int}}}
```

The `*`-form of `\ref` was used to turn off the redundant link creation. (hyperref would normally make the `\ref` macro into a link.)

▶ An 'Exercise' that is also a hypertext link appears in the default color green; if an 'Exercise' is not a link, it appears in blue. (The word 'Exercise' is not a link if it is a exercise with parts, or if the `nosolutions` options is used. Finally, if the `web` option `forpaper` is used, color is turned off and 'Exercise' appears in black.

▶ **Caveat:** There is one problem you might watch for. There is an optional argument to the `solution` environment. When LaTeX searches the source looking for the optional parameter, which may not exist, it expands macros looking for a '`[`'. This cases problem when you have a solution that begins with a math display environment and LaTeX prematurely expands such an environment.

Exercise 2. Write an equation of a line that crosses the x- and y-axes at 1.

To prevent LaTeX errors that will stop the compilation, just place a `\relax` prior to the math environment. The code for the previous

exercise is

```
\begin{exercise}
Write an equation of a line that crosses
the $x$- and $y$-axes at 1.
\begin{solution}
\relax\begin{equation*}
                    \boxed{x+y=1}
\end{equation*}
\end{solution}
\end{exercise}
```

This is only necessary if the solution does not begin with text.

- **An exercise with Parts**

There is a *-option with the exercise environment, using it signals the presence of a multiple part exercise question. The syntax is as follows:

```
\begin{exercise}*              % *-option
Preamble for your multi-parted question.
\begin{parts}                  % begin listing of the parts
\item First question.
\begin{solution}
```

```
Solution to first question.
\end{solution}
...
...
\item Final question.
\begin{solution}
Solution to the final question.
\end{solution}
\end{parts}                    % end listing of parts
\end{exercise}
```

The following exercise illustrates this option. This example appears in the demo file webeqtst.tex.

EXERCISE 3. Suppose a particle is moving along the s-axis, and that its position at any time t is given by $s = t^2 - 5t + 1$.

(a) Find the velocity, v, of the particle at any time t.

(b) Find the acceleration, a, of the particle at any time t.

There is also an option for listing multiparts question in tabular form.

EXERCISE 4. Simplify each of the following expressions in the complex

number system. *Note*: $\bar{z}$ is the conjugate of z; $\operatorname{Re} z$ is the real part of z and $\operatorname{Im} z$ is the imaginary part of z.

(a) i^2 (b) i^3

(c) $z + \bar{z}$ (d) $1/z$

The syntax is the same as an exercise with multiparts

```
\begin{exercise}* % <- star indicates multipart
Simplify each...
\begin{parts}[2]  % <- optional argument indicates tabular
\item $i^2$
\begin{solution} $i^2 = -1$ \end{solution}
&
\item $i^3$ \begin{solution} $i^3 = i i^2 = -i$\end{solution}
\\
\item $z+\bar z$
\begin{solution} $z+\bar z=\operatorname{Re} z$\end{solution}
&
...
\end{solution}
\end{parts}
\end{exercise}
```

▶ This problem style does not obey the `solutionsafter` option. (See

'The solutionsafter option' on page 72).

▶ The sample file webeqtst.tex contains this particular example.

5.2. Options of the exercise Environment

- **Leaving Vertical Space instead of a Solution**

The exercise environment can be used for test construction. Initially, you may want to pose a questions and leave space beneath for the student to write in an answer.

The solutions environment has an optional parameter for insert a vertical space.

```
\begin{exercise}
This is the question.
\begin{solution}[1in]    % <-- optional vertical skip
This is the solution.
\end{solution}
\end{exercise}
```

This vertical space only appears when the nosolutions option is in effect.

Within the context of test construction, write the test (including the solutions) then publish it with the **nosolutions** option (leaving vertical spaces as appropriate) then publish the key with the solutionsafter option. (If **solutionsafter** and **nosolutions** both appear in the option list, **solutionsafter** overrides **nosolutions**.)

▶ The optional parameter for the solution is ignored for exercises with parts having a tabular format (Example 4 is an example of a tabular multipart exercise).

- **Hiding some Solutions**

A subset of the solutions can be hidden by using the 'h' option. This option is an option of the **exercise** environment, as well as an option of **\item**, when there is an exercise with parts. For example, the following code

```
\begin{exercise}[h]  % <- hide solution
Give an example of a set that is \textit{clopen}.
\begin{solution}
The real number line is both closed and open in the
usual topology of the real line.
```

```
\end{solution}
\end{exercise}
```

yields the exercise

EXERCISE 5. Give an example of a set that is *clopen*.

Notice that there is no hypertext link to the solution; indeed, the solution was not even written to the `\jobname.sol` file.

The 'h' option works with exercises with parts as well, just apply the 'h' option to the `\item`:

```
\begin{exercise}*
A particle has position $s=t^2 - 5t + 1$ at time $t$.
\begin{parts}

\item Find the velocity, $v$, at time $t$.
\begin{solution}
$v = 2t-5$.
\end{solution}

% This solution will not be included in the solutions
% section at the end of the document.
\item[h] Find the acceleration, $a$, at time $t$.
```

```
\begin{solution}
$a = 2$.
\end{solution}
\end{parts}
\end{exercise}
```

The results of this code follow:

EXERCISE 6. A particle has position $s = t^2 - 5t + 1$ at time t.

(a) Find the velocity, v, at time t.

(b) Find the acceleration, a, at time t.

Part (a) is hypertext linked to its solution, whereas part (b) is blue, indicating there is no link there.

▶ Multipart exercises in the tabular format behave the same way; use `\item[h]` to "hide" a solution.

▶ There is also an 'H' option as well. Specifying 'H' also hides the solutions. See the next two sections for a discussion of revealing the solutions marked by either 'h' or 'H' to understand the distinction between the two.

- **The nohiddensolutions Option**

Hidden solutions can be included in the document by either removing the 'h' option everywhere and re-LaTeXing, or by simply using the nohiddensolutions of exerquiz.

```
\usepackage[nohiddensolutions]{exerquiz}
```

This option overrides the local 'h' option throughout the document.

▶ When the solutionsafter option of exerquiz is involked, the hidden solutions are also revealed. To keep the solutions hidden, in this case, you should use 'H' option instead of 'h'. See the next section.

- **The noHiddensolutions Option**

In addition to the 'h', you can also use the 'H' option with exercises. The solution will be hidden with 'H', but will not be revealed when either the nohiddensolutions or the solutionsafter options are used.

The 'H' option can be overridden by using the noHiddensolutions of exerquiz.

```
\usepackage[noHiddensolutions]{exerquiz}
```

This option overrides the local 'h' option throughout the document.

- **The counter for the exercise environment**

The counter for the exercise environment is eqexno, and will number your exercises consecutively throughout the document. Should you want the counter to be reset after each section, place in the preamble of your document the following lines:

```
\makeatletter
\@addtoreset{eqexno}{section}
\makeatother
```

- **The nosolutions option**

Some educators may initially want to post a series of exercises on the Web without the solutions. Then, at a later date, repost the exercises with the solutions included. For this application there is the nosolutions option for the exerquiz package.

```
\documentclass{article}
\usepackage[pdftex]{web}  % dvipsone, dvips or dvipdfm
\usepackage[nosolutions]{exerquiz}
```

For this kind of application, it might make sense to publish the exercises with the `forpaper` option.

- **The `solutionsafter` option**

For additional flexibility with how you want the solutions to the exercises presented, there is a `solutionsafter` option with exerquiz. Should you invoke this option

```
\documentclass{article}
\usepackage[dvipsone]{web}  % dvips or pdftex
\usepackage[solutionsafter]{exerquiz}
```

the solutions to the exercises appear just *after* the exercise question. For example

EXERCISE 7. Let V be a vector space, show that the zero vector, $\mathbf{0}$, is unique.

Solution: Let $\mathbf{0}'$ be a vector that satisfies the axiom of being a zero of the vector space V. We want to show $\mathbf{0} = \mathbf{0}'$. Since $\mathbf{0}$ is a zero,

we have $\mathbf{0} + \mathbf{0}' = \mathbf{0}'$. But we are assuming $\mathbf{0}'$ is a zero vector as well, hence, $\mathbf{0}' + \mathbf{0} = \mathbf{0}$. Finally,

$$\mathbf{0}' = \mathbf{0} + \mathbf{0}' = \mathbf{0}' + \mathbf{0} = \mathbf{0}$$

and this completes the proof. Exercise 7

The option `solutionsafter` is global; all exercises will be typeset this way—unless you change it within the document using the macros `\SolutionsAfter` and `\SolutionsAtEnd`. This manual was typeset without the `solutionsafter` option. The above example was typeset as follows:

```
\SolutionsAfter  % show solution following exercise
\begin{exercise}
Let $V$ be a vector space, show ...
\begin{solution}
............
\end{solution}
\end{exercise}
\SolutionsAtEnd  % turn back on solutions at of document
```

Normally, a typical document might have all solutions at the end of the document (the default behavior), or all solutions following

each exercise (solutionsafter option). Mixtures of these two types can be obtained by using the two commands \SolutionsAfter and \SolutionsAtEnd.

This feature might be an easy way of typsetting examples. See the paragraph 'Redesigning the exercise Environment' on page 75 for an example of setting up an example environment.

▶ The solutionsafter option has no effect on multipart exercises in *tabular form*; I haven't been able to find a convenient way of displaying the solutions after the questions when the questions are in tabular form.

▶ See the files webeqtst.pdf and hw02.pdf (and their source files) for examples.

- **Moving the Solution Set**

The solution set, by default, comes last in the file. You can move its positioning by including the command \includeexersolutions at any point *after* the last exercise. You'll note, that I have moved the solutions in this file before the References section, as indicated, for

example, by its position in the table of contents.

5.3. Redesigning the exercise Environment

You can customize the exercise environment to suite your own needs. To customize, you need to change some or all of the following six commands. In the listing below, the LaTeX definition of each follows a short description.

1. \exlabel: This command expands to the name of the exercise label, the default string is 'Exercise'.

```
\newcommand\exlabel{Exercise}
```

2. \exlabelformat: Typesets the exercise label; use it to introduce additional type style such as boldface, italic, small caps, etc.

```
\newcommand\exlabelformat{%
      {\scshape\exlabel\ \theeqexno.}}
```

3. \exlabelsol: Expands to the name of the exercise label in the solutions section. Usually its value is the same as \exlabelsol.

```
\newcommand\exlabelsol{\exlabel}
```

4. `\exsllabelformat`: The format of the solutions label, the default is '`\bfseries\exlabel`'.

```
\newcommand\exsllabelformat
     {\noexpand\textbf{\exlabelsol\ \theeqexno.}}
```

5. `\exrtnlabelformat`: This is the label you click on to return from the solution of the exercise.

```
\newcommand\exrtnlabelformat{\exlabelsol\ \theeqexno}
```

6. `\exsectitle`: The section title of the solutions to the exercises.

```
\newcommand\exsectitle{Solutions to \exlabel s}
```

7. `\exsecrunhead`: The running header for the solution section for the exercises.

```
\newcommand\exsecrunhead{\exsectitle}
```

▶ The counter `eqexno` is used to count exercises. When the `exercise` environment starts, this counter is incremented. Normally, the values of this counter figures into the definitions of `\exlabelformat`, `\exsllabelformat` and `\exrtnlabelformat`. Still, the use of `eqexno`

is optional; for example, you might want to state a problem just as 'Special Exercise', without an associated exercise number.

Below is an example of redefining the `exercise` environment. We define a `problem` environment based on the `exercise` environment.

```
\newenvironment{problem}{%
\renewcommand\exlabel{Problem}
\renewcommand\exlabelformat{\textbf{\exlabel\ \theeqexno.}}
\renewcommand\exsllabelformat
   {\noexpand\textbf{\exlabel\ \theeqexno}}
\renewcommand\exrtnlabelformat{$\blacktriangleleft$}
\renewcommand\exsecrunhead{\exsectitle}
\begin{exercise}}%
{\end{exercise}}
```

See any standard LaTeX reference on how to define a new environment, for example [3].

Here is an example of the new `problem` environment:

Problem 8. This is a question.

The code for this problem was simply:

```
\begin{problem}
```

```
This is a question.
\begin{solution}
This is the solution.
\end{solution}
\end{problem}
```

▶ Two of these commands must be handled with special care, they are `\exsllabelformat` and `\exrtnlabelformat`; formatting such as `\textbf` or `\scseries` must be preceded by a `\noexpand`. These commands are written to a file, and must be prevented from expanding.

When you use the `exercise` environment, the counter `eqexno` is automatically incremented by default. The `exercise` does have an optional argument for inserting your own counter.

```
\begin{exercise}[<ctr>]
.......................
\end{exercise}
```

Where `<ctr>` is a counter already defined. This option is useful if you want to use the `exercise` environment to create a new environment with its own numbering scheme, as the following example illustrates.

This example demonstrates how to define an `example` environment with its own counter. For examples, we don't want the solutions to appear at the end of the file, so we'll use `\SolutionsAfter` and `\SolutionsAtEnd`. All changes are local.

```
% put a counter in preamble
\newcounter{exampleno}
\newenvironment{example}{%
\renewcommand\exlabel{Example}
\renewcommand\exlabelformat
  {\textbf{\exlabel\ \theexampleno.}}
\renewcommand\exrtnlabelformat{$\square$}
\SolutionsAfter
\begin{exercise}[exampleno]}%
{\end{exercise}
\SolutionsAtEnd}
```

Now we simply type

```
\begin{example}
What is $2+2$?
\begin{solution}
It is well known that $2+2=4$.
\end{solution}
\end{example}
```

to obtain

Example 1. What is $2 + 2$?

Solution: It is well known that $2 + 2 = 4$. □

Example 2. What is $2 + 2$?

Solution: It is well known that $2 + 2 = 4$. □

The changes are local to the new `example` environment. If we have another exercise, we get a correctly numbered exercise.

Exercise 9. What is $2 + 2$?

▶ The command `\exsolafter` typesets the solution label to the exercise in the case the `solutionsafter` option is in effect. The default value of `\exsolafter` is `\textit{Solution}:` You can redefine it as follows:

```
\renewcommand\exsolafter{\textsl{L\"osung}:}
```

This redefinition yields:

Example 3. What is $2 + 2$?

Lösung: It is well known that $2 + 2 = 4$. □

- There is a special option to the `exercise` environment as well,

```
\begin{exercise}[0]
.......................
\end{exercise}
```

When the optional argument is `0` rather than a counter. In this case, no counter is associated with the environment. For example,

```
\newenvironment{project}{%
\renewcommand\exlabel{Project}
\renewcommand\exlabelformat{\textbf{\exlabel. }}
\renewcommand\exsllabelformat
   {\noexpand\textbf{\exlabel\ Hint:}}
\renewcommand\exrtnlabelformat{$\blacktriangleleft$}
\begin{exercise}[0]}%
{\end{exercise}}
```

Thus, we obtain,

Project. Find a shorter proof of Fermat's Last Theorem. Do not look at the project hints until you have finished the project.

The code:

```
\begin{project}
Find a shorter proof of \textsc{Fermat's Last Theorem}. Do not
look at the project hints until you have finished the project.
\begin{solution}
There, you didn't need my help after all.
\end{solution}
\end{project}
```

Note that the solutions are typeset at the end of the file in the 'Solutions to Exercises' section. At this time, there is no feature for sorting out these different types of environments; they are all `exercise` environments, which is what they are.

▶ Finally, see the sample file `hw01.tex` that illustrates how to change all the labels. The file also demonstrates how web and exerquiz can be used to post problems on the Internet, or on paper, with or without solutions included.

6. The shortquiz Environment

The **shortquiz** environment is used to create multiple choice question and math/text fill-in questions with immediate response. The environments allow redefinition to customize the look you the quizzes. (See the paragraph entitled 'Redesigning the shortquiz Environment' on page 95.)

The discussion of math and text fill-in questions is post-phoned to Section 8, entitled Objective Style Questions.

The presentation of the answers will either be in a **list** or a **tabular** environment, depending on the parameter **num_cols**. (A **list** is used if **num_col** is set to 1.)

6.1. Basic Usage

The syntax for the environment (**tabular** version) is as follows:

```
\begin{shortquiz}                    % begin shortquiz
...Question goes here...
\begin{answers}{num_cols}            % begin proposed answers
...
\Ans0 <an incorrect answer> &        % a wrong answer
```

```
...
\Ans1 <a correct answer> &          % the right answer
...
\end{answers}                       % end listing of answers
\end{shortquiz}                     % end shortquiz
```

The parameter num_cols is the number of columns you want to typeset your multiple choice responses in, which is a tabular environment. *Note*: If num_cols is 1, a list environment is created rather than a tabular.

This type of quiz is suitable as a quiz in-line question of the reader, perhaps after explaining some concept. Quizzes can be used to direct the reader's attention to an important point.

▶ Here is an example of the shortquiz environment. Responses are graded without comment using JavaScript.

Quiz Which of the following is the $\frac{d}{dx}\sin(x^3)$?

(a) $\sin(3x^2)$ (b) $\cos(x^3)$ (c) $3x^2\cos(x^3)$ (d) $3x^2\cos(3x^2)$

The verbatim listing follows:

```
\begin{shortquiz}          % begin shortquiz environment
Which of the following is the $\dfrac{d}{dx}{\sin(x^3)}$?
\begin{answers}{4}         % 4 columns of answers
  \Ans0 $\sin(3x^2)$ &     % \Ans0 is a false answer
  \Ans0 $\cos(x^3)$ &
  \Ans1 $3x^2\cos(x^3)$ & % \Ans1 is the correct answer
  \Ans0 $3x^2\cos(3x^2)$
\end{answers}              % end answers environment
\end{shortquiz}            % end shortquiz environment
```

If num_cols, the argument of the answers environment is 1, a list environment is created; otherwise, the answers environment uses a tabular with p{<width>} to set up the columns. The \parboxes are typeset ragged right.

▶ Below is a two-column example in which the posed alternatives are rather long. The answers environment produces is a nicely aligned set of paragraphs.

Quiz Which of the following best describes Augustin Cauchy?

(a) He developed the Calculus while his University was closed for the plague.

(b) Given credit for first using the functional notation $f(x)$.

(c) He created the "bell-shaped curve" and first used the method of least squares.

(d) He first formulated a precise definition of the limit and continuity of a function.

(e) Gave a rigorous definition of the definite integral—an integral that now bears his name.

(f) His notation for the derivative and the integral is used even to this day.

Here is the same example in which the `num_cols` is set to 1; in this case, a `list` environment is used.

Quiz Which of the following best describes Augustin Cauchy?

(a) He developed the Calculus while his University was closed for the plague.

(b) Given credit for first using the functional notation $f(x)$.

(c) He created the "bell-shaped curve" and first used the method of least squares.

(d) He first formulated a precise definition of the limit and continuity of a function.

(e) Gave a rigorous definition of the definite integral—an integral that now bears his name.

(f) His notation for the derivative and the integral is used even to this day.

▶ See the sample files webeqtst.tex and qz01.tex for examples. The later file gives examples of how to redefine some of the standard shortquiz labels.

- **shortquiz with Solutions**

Another type of quiz that is easy to implement in PDF is the multiple choice quiz with immediate response with solution given. This too is a shortquiz environment:

```
\begin{shortquiz}
...Question goes here...
\begin{answers}[<name>]{<num_cols>}
...
```

```
\Ans0 <an incorrect answer> &
...
\Ans1 <a correct answer> &
...
\end{answers}
\begin{solution}
...Solution to correct answer goes here...
\end{solution}
\end{shortquiz}
```

The <name> is a name used to create a hypertext jump to the solution; <name> will be the "named destination." As before, <num_cols> is the number of columns to typeset the answers in.

The following example illustrates the quiz with solution.

Quiz Define a function $f(s) = 4s^3$ and another function $F(t) = t^4$. Is F an antiderivative of f?

(a) Yes (b) No

The verbatim listing:

```
\begin{shortquiz}
Define a function $f(s)=4s^3$ and another
```

```
function $F(t)=t^4$. Is $F$ an antiderivative of $f$?
\begin{answers}[quiz:anti]{4}
\Ans1 Yes &\Ans0 No
\end{answers}
\begin{solution}
The answer is ‘Yes’.  The definition requires that
$$
          F’(x) = f(x) \quad\text{for all $x$,}
$$
well, let’s check it out.
.........................
.........................
Therefore,
$$
       F’(x) = 4x^3 = f(x)\quad\text{for all $x$,}
$$
as required by the definition.
\end{solution}
\end{shortquiz}
```

• The `questions` Environment

The `questions` environment was designed to work with the `quiz` environment—taken up in Section 7 below—but it works equally well with `shortquiz`.

Using the `questions` environment, quizzes defined by `shortquiz`, with/without solutions, can be mixed together and combined to make a "mini-quiz". For example,

Quiz Determine the LCD for each of the following.

1. $\dfrac{3x}{2y^2z^3} - \dfrac{2}{xy^3z^2}$.

(a) LCD $= 2xy^5z^5$
(b) LCD $= 2y^3z^3$
(c) LCD $= 2xy^3z^3$
(d) LCD $= 2xy^3z^5$

2. $\dfrac{x+y}{3x^{3/2}y^2} - \dfrac{x^2+y^2}{6xy^4}$.

(a) LCD $= 18x^{3/2}y^4$
(b) LCD $= 6x^{3/2}y^4$
(c) LCD $= 18xy^4$
(d) LCD $= 6xy^4$

The first question is given without a solution, the second has a

solution attached to it. An abbreviate verbatim listing follows.

```
\begin{shortquiz}
Determine the LCD for each of the following.
\begin{questions}
\item  $\dfrac{3x}{2y^2z^3}-\dfrac2{xy^3 z^2}$.
\begin{answers}2
...
\end{answers}
\item  $\dfrac{x+y}{3 x^{3/2}y^2}
          -\dfrac{x^2+y^2}{6 x y^4}$.
\begin{answers}[quiz:LCB]2
...
\end{answers}
\begin{solution}
If you erred on this one, ... ...
\end{solution}
\end{questions}
\end{shortquiz}
```

6.2. Options of the shortquiz Environment

- **The forpaper option**

The forpaper option has already been described. The solutions to a shortquiz questions are not typeset on separate pages, but are separated by a \medskip.

Following up on the pretest angle first discussed in an earlier paragraph, Redesigning the shortquiz Environment, page 95, a single document can be constructed that can be published on-line, or published for paper distribution. This feature may be useful to some educators.

By the way, if you want to create a series of multiple choice questions with solutions, you must make up a lot of named destinations (the optional argument of the answers environment). Alternately, you can let LaTeX assign the names for you, which provides for you a uniform naming system. You can use questionno to do this:

```
\begin{shortquiz} Answer each, then look at solutions.
  \begin{questions}
    \item  ...
      \begin{answers}[quiz:\thequestionno]{4}
```

```
      ...
    \end{answers}
    \begin{solution}
      ...
    \end{solution}
  \item  ...
    \begin{answers}[quiz:\thequestionno]{4}
      ...
    \end{answers}
    \begin{solution}
    ...
    \end{solution}
\end{questions}
\end{shortquiz}
```

- **The `solutionsafter` Option**

The `solutionsafter` option works as described for the `exercise` environment. The option just sets a boolean switch. This switch can be controlled locally with the macros `\SolutionsAfter` and `\`. Here is a simple example.

Quiz In what year did Columbus sail the ocean blue?

(a) 1490 (b) 1491 (c) 1492 (d) 1493

Solution: Columbus sailed the ocean blue in 1492. Some say he discovered San Salvatore, others say he first sited Cat Island in the Bahamas.

End Quiz

Here, I have surrounded the `shortquiz` environment with the command `\SolutionsAfter` before the environment, and with the command `\SolutionsAtEnd` just after.

This option may be useful in publishing an answer key to a multiple choice quiz. The quiz and solutions can be created together. The quiz can be published, then later, the quiz with complete solutions.

- **The `proofing` Option**

For proofreading, use the `proofing` option of exerquiz.

```
\usepackage[proofing]{exerquiz}
```

When used, a symbol, defined by the command `\proofingsymbol`, will mark the correct answers, as defined in your source file. The command `\proofingsymbol` can be redefined, its definition is

```
\newcommand\proofingsymbol{\textcolor{webgreen}{$\bullet$}}
```

This option works for the quiz environment defined below (page 98), as well.

- **Moving the Solution Set**

The solution set, by default, comes last in the file. You can move its positioning by including the command \includequizsolutions at any point *after* the last exercise. You'll note, that I have moved the solutions in this file before the References section, as indicated, for example, by its position in the table of contents.

6.3. Redesigning the shortquiz Environment

You can temporarily change the title for the shortquiz environment by redefining the macro \sqlabel; for example, the default definition of this macro is

```
\newcommand\sqlabel{\textcolor{red}{Quiz.}}
```

The syntax for redefining \sqlabel is

```
\renewcommand\sqlabel{...new code goes here...}
```

You can redefine the *default* label as well; the default label is the title label that **shortquiz** uses when **\sqlabel** is *not present*. The default label is **\eq@sqlabel** and must be redefined using the macro **\renewcommand**. The best place for this to be done is the preamble. The syntax:

```
\makeatletter    % make ‘at’=@ a normal letter
\renewcommand\eq@sqlabel{...new code goes here...}
\makeatother     % make ‘at’=@ something special(other)
```

To change the entire document to use ‘Exam’ instead of ‘Quiz’, make the following changes in the preamble:

```
\makeatletter
% change default quiz title to ’Exam’
\renewcommand\eq@sqlabel{\textcolor{red}{Exam.}}
% change quiz solutions return label
\renewcommand\eq@sqslrtnlabel{End Exam}
% change solutions label
\renewcommand\eq@sqsllabel{%
   \string\textbf{Solution to Exam:}}
\renewcommand\eq@sqslsectitle{Solutions to Exams}
% change default running header for solutions
\renewcommand\eq@qslsecrunhead{Solutions to Exams}
```

\makeatother

▶ The above commands are 'global'—they are in effect throughout the entire document. You can temporarily change these labels using the \sqlabel, \sqslrtnlabel, \sqsllabel and \sqslsectitle. Note that you cannot temporary change \eq@qslsecrunhead, the running label—this should be set in the preamble.

Should you want to make a series of multiple choice questions (using the questions environment) and combine them into a sort of review or pretest, a useful idea would be to number the solutions. The counter that maintain the question number is called questionno. You can then, for example, define

```
\renewcommand\eq@sqsllabel{%
   \string\textbf{Solution to Question \thequestionno:}}
```

▶ See the sample files webeqtst.tex and qz01.tex for examples. The later file gives examples of how to redefine some of the standard shortquiz labels.

7. The quiz Environment

Use the **quiz** environment to create graded quizzes. In this case, several (many) questions are bundled together. The student takes the quiz and responses are recorded by JavaScript. Upon completion of the quiz, the total score is reported to the student.

The **quiz** environment can generate multiple choice questions and math/text fill-in questions. The discussion of math and text fill-in questions is post-phoned to Section 8 on page 123

There are two types of quizzes, the link-style and form-style. In Section 7.2, we see that the **quiz** environment can also correct the quizzes.

The **quiz** environment consists of a series of nested environments. Inside the **quiz** environment is the **questions** environment (an enumerated list), and within that environment is the **answers** environment. Symbolically, we can express this as

$$\texttt{quiz} \supseteq \texttt{questions} \supseteq \texttt{answers}$$

The term 'answers' is, perhaps, not sufficiently descriptive; 'alternatives' would be more appropriate, but it requires more typing. :-)

▶ The answers environment requires one parameter, the num_cols. If num_cols is 1, a list environment is created; otherwise, a tabular environment is used.

This (tabular) environment has the following syntax:

```
\begin{quiz}{quizfieldname}
The preamble to the questions goes here.
\begin{questions}
\item State first question....
\begin{answers}4 % <- num_cols = 4
\Ans0 ... &\Ans1 ... &\Ans0 ... &\Ans0 ...
\end{answers}
...
\item n th question....
\begin{answers}4                        % <- 4 column format
\Ans0 ... &\Ans1 ... &\Ans0 ... &\Ans0 ...
\end{answers}
\end{questions}
\end{quiz}
```

▶ Following the quiz, or anywhere in the document, place the macro \ScoreField, defined in exerquiz, to display the results of the quiz:

```
\ScoreField{quizfieldname}
```

Important. The value of the parameter of the macro `\ScoreField` must match the `quizfieldname` defined in the argument of the `quiz` environment.

▶ There is a convenience macro, `\currQuiz`, that holds the name of the current quiz. Thus, we could have instead typed:

```
\ScoreField\currQuiz
```

Read the paragraph entitled 'The Score Field' on page 121 for more details on this macro.

7.1. Basic Usage

In this section we discuss the two basic `quiz` styles: Link-Style Quiz and Form-Style Quiz.

A paragraph is devoted to some modification that can be made to the beginning and end of the quiz. In addition, a `proofing` option is also described.

- **Link-Style Quiz**

This style uses links to record the choices to the alternatives. The link method takes up less space in the pdf file than does the form-style.

Below is an example of a link-style quiz. Instructions should be given to guide the student in operating the quiz correctly.

Instructions. You must click on 'Begin Quiz' to initialize the quiz. Not doing so, brings forth an error message. When finished, click on 'End Quiz'.

Begin Quiz Using the discriminant, $b^2 - 4ac$, respond to each of the following questions.

1. Is the quadratic polynomial $x^2 - 4x + 3$ irreducible?

(a) Yes (b) No

2. Is the quadratic polynomial $2x^2 - 4x + 3$ irreducible?

(a) Yes (b) No

3. How many solutions does the equation $2x^2 - 3x - 2 = 0$ have?

(a) none (b) one (c) two

End Quiz

Score:

▶ While you are taking the test, and before you click on 'End Quiz', you can change your answers. A message box comes out, gives you your original choice, and asks you whether you really want to change your answer.

```
\begin{quiz}{qz:discr-1}  % qz:discr=quiz field name
Using the discriminant, $b^2-4ac$, respond to each of the
following questions.
\begin{questions}
\item Is the quadratic polynomial $x^2-4x + 3$ irreducible?
\begin{answers}4
\Ans0 Yes &\Ans1 No
\end{answers}
\item Is the quadratic polynomial $2x^2-4x+3$ irreducible?
\begin{answers}4
\Ans1 Yes &\Ans0 No
\end{answers}
\item How many solutions does the equation $2x^2-3x-2=0$ have?
\begin{answers}4
\Ans0 none &\Ans0 one &\Ans1 two
```

```
\end{answers}
\end{questions}
\end{quiz}\par
\ScoreField\currQuiz % matching quiz field name
```

▶ The convenience text macro, `\currQuiz`, contains the name of the the current quiz. This macro can be used as the argument of `\ScoreField`.

• Form-Style Quiz

You may be thinking that such a quiz format—one in which the student cannot see the choices made—is not very good. It is perhaps adequate for two or three quick questions. For a longer quiz format, one would like to see a "checkbox" format. A quiz with a checkbox format can be obtained using the *-form of the `quiz` environment:

```
\begin{quiz}*{quizfieldname}
...same format as before...
\end{quiz}
```

Here is the same sample quiz with the form-style option. The only change in the code is the insertion of the *-option.

Instructions. You must click on 'Begin Quiz' to initialize the quiz. Not doing so, brings forth an error message. When finished, click on 'End Quiz'.

Begin Quiz Using the discriminant, $b^2 - 4ac$, respond to each of the following questions.

1. Is the quadratic polynomial $x^2 - 4x + 3$ irreducible?

☐ Yes ☐ No

2. Is the quadratic polynomial $2x^2 - 4x + 3$ irreducible?

☐ Yes ☐ No

3. How many solutions does the equation $2x^2 - 3x - 2 = 0$ have?

☐ none ☐ one ☐ two

End Quiz

Score:

▶ Before completing the quiz, a student can easily change alternatives.

▶ This type is more suitable for longer quizzes. The choices student make are visually recorded for the student to review and change before clicking on 'End Quiz'. A partial verbatim listing:

```
\begin{quiz}*{qz:discr-f}
Using the discriminant, $b^2-4ac$, respond to each of the
following questions.
\begin{questions}
.............
.............
\end{questions}
\end{quiz}\par
\ScoreField{qz:discr-f}
```

▶ See the sample files webeqtst.tex and qz02.tex for examples. The later file gives examples of how to customize quiz.

- **Overriding the 'quiztype' Parameter**

You can globally declare that all quizzes to be a link-type or form-type by using the command \quiztype. Placing \quiztype{f} in the preamble (or prior to any quiz) will cause all quizzes following

that command to be form-type quizzes. Similarly, `\quiztype{l}` will produce all link-type quizzes.

The command `\quiztype` causes the `quiz` environment to ignore the first optional parameter (the '`*`'). You can make the environment obey this optional parameter by using `\defaultquiztype`.

The sample file quizpts.tex illustrates these collections of macros.

• The BeginQuiz and EndQuiz Form Buttons

The default setup the the `quiz` environment is to have hypertext links for the 'Begin Quiz' and 'End Quiz'. You can also redefine this linking and use a form button instead Prior to your quiz, use the following code, if desired.

```
\useBeginQuizButton
\useEndQuizButton
```

Begin Quiz Answer each of the following. Passing is 100%.

1. Who created TEX?

(a) Knuth (b) Lamport (c) Carlisle (d) Rahtz

2. Who originally wrote LaTeX?

(a) Knuth (b) Lamport (c) Carlisle (d) Rahtz

End Quiz Score:

Revert back to link-style as follows:

```
\useBeginQuizLink
\useEndQuizLink
```

The commands `\useBeginQuizButton` and `\useEndQuizButton` each have an optional argument that can be used to modify the appearance of the buttons. These buttons work with Link- or Form-type quizzes and are independently customizable, see For details, see the section entitled The 'Correction' Button. For an example, see the sample file `quizpts.tex`

- **The `proofing` Option**

For proofreading, use the `proofing` option of exerquiz.

```
\usepackage[proofing]{exerquiz}
```

When used, a symbol, defined by the command \proofingsymbol, will mark the correct answers, as defined in your source file. The command \proofingsymbol can be redefined, its definition is

```
\newcommand\proofingsymbol{\textcolor{webgreen}{$\bullet$}}
```

This option works for the shortquiz environments defined above (page 83), as well.

- **Setting the Threshold**

The default behavior of the quiz environment is that a student can begin the quiz and finish the quiz without answering any or all of the questions. This is called a lowThreshold and is the default behavior.

The document author can set a highThreshold be re-defining the \minQuizResp macro. The default defintion is

```
\newcommand\minQuizResp{lowThreshold}
```

However, if you make the definition

```
\renewcommand\minQuizResp{highThreshold}
```

the student is required to answer all the questions of a quiz.

Actually, lowThreshold and highThreshold are JavaScript functions that are called when the "End Quiz" button is clicked. If the threshold is not met, an alert box appears informing the user of this.

The document author can write a custom threshold function and place its name in the \minQuizResp macro. See the exerquiz source code for the highThreshold() function for an example of how to do this.

7.2. Correcting the Quizzes with JavaScript

Beginning with exerquiz, version 1.2, you can now correct quizzes created by the quiz environment. To correct the quizzes, simply include an additional element into your quiz, a correction button. The correction button is installed using the macro \eqButton.

The following is a link-style quiz.

Instructions: Click on 'Begin Quiz' to initialize the quiz. When finished, click on 'End Quiz'. Then, click on the 'Correct' button.

Begin Quiz Using the discriminant, $b^2 - 4ac$, respond to each of the

following questions.

1. Is the quadratic polynomial $x^2 - 4x + 3$ irreducible?
   (a) Yes (b) No
2. Is the quadratic polynomial $2x^2 - 4x + 3$ irreducible?
   (a) Yes (b) No
3. How many solutions does the equation $2x^2 - 3x - 2 = 0$ have?
   (a) none (b) one (c) two

End Quiz

Score: Correct

Legend: A ✔ indicates a correct response; a ✘, indicates an incorrect response, in this case, the correct answer is marked with a ●.

A partial verbatim listing of this quiz follows:

```
\begin{quiz}{qz:discr1-1} Using the discriminant, $b^2-4ac$,
respond to each of the following questions.
\begin{questions}
.........................
```

```
.............................
.............................
\end{questions}
\end{quiz}

\ScoreField{qz:discr1-l}\eqButton{qz:discr1-l}
```

▶ The macro `\eqButton` is used to create a nice "correction" button. JavaScript is used to correct the quiz. The only required argument is the field label that uniquely defines the field in which the total score is placed. See the section entitled 'The 'Correction' Button' on page 118 for more details on how to use this macro.

▶ The `\eqButton` will not work until the user has clicked on 'End Quiz'. The user can re-take the quiz simply by clicking on 'Begin Quiz', the form fields and JavaScript variables will be cleared.

▶ It is possible to take this form data and submit it to a CGI script for processing (The data can be saved to a database, for example.) However, there is no built-in capability for this in the exerquiz package.

The same quiz can be written in form-style simply by inserting the

*-option.

Instructions. You must click on 'Begin Quiz' to initialize the quiz. Not doing so, brings forth an error message. When finished, click on 'End Quiz'.

Begin Quiz Using the discriminant, $b^2 - 4ac$, respond to each of the following questions.

1. Is the quadratic polynomial $x^2 - 4x + 3$ irreducible?

☐ Yes ☐ No

2. Is the quadratic polynomial $2x^2 - 4x + 3$ irreducible?

☐ Yes ☐ No

3. How many solutions does the equation $2x^2 - 3x - 2 = 0$ have?

☐ none ☐ one ☐ two

End Quiz Score: Correct

▶ In the partial verbatim listing that follows, notice the field name

as been changed from qz:discr1-l to qz:discr1-f. The different quizzes must have a unique field name.

```
\begin{quiz}*{qz:discr1-f} Using the discriminant, $b^2-4ac$,
respond to each of the following questions.
\begin{questions}
.............................
.............................
.............................
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

▶ Notice that in this example, the \ScoreField and the \eqButton are positioned following the 'End Quiz'; this makes the design more compact and nicer looking.

- **The nocorrections Option**

Including the corrections adds quite a bit more JavaScript code to the .pdf document, this feature is 'on' by default. If you have a document in which you do not want to have the option of offering corrected quizzes, then just specify nocorrections is the option list of exerquiz.

There are also a couple of macros you can use to override the option switch: `\CorrectionsOn` and `\CorrectionsOff`. Each remains in affect until the other is invoked.

7.3. Quizzes with Solutions

In addition to scoring and marking the quizzes, you can also (optionally) provide solutions as well. To enter a solution to a multiple choice question, use a `solution` environment, and attached a named destination to the `answers` environment. A partial verbatim listing of the follows the next example.

Begin Quiz Answer each of the following. Passing is 100%.

1. Who created TEX?

☐ Knuth ☐ Lamport ☐ Carlisle ☐ Rahtz

2. Who originally wrote LATEX?

☐ Knuth ☐ Lamport ☐ Carlisle ☐ Rahtz

End Quiz Score: Correct

After the quiz is completed and the corrections button is pressed, the corrections appear. The correct answer has a green filled circle or a green check; this circle is now outlined by a green rectangle to indicate that this is a link to the solution. Click on the green dot and jump to the solution!

Solution do not have to appear. Some problems can have solutions, while others do not. The ones with the solutions have the green boundary to indicate a link to the solution.

Here is a partial listing of the above example.

```
\begin{quiz}*{qz:TeX-1} Answer each of the following.
Passing is 100\%.
\begin{questions}
\item Who created \TeX?
\begin{answers}[knuth]4
\Ans1 Knuth &\Ans0 Lamport &\Ans0 Carlisle &\Ans0 Rahtz
\end{answers}
\begin{solution}
Yes, Donald Knuth was the creator of \TeX.
\end{solution}
....
\end{questions}
```

```
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

▶ Notice that in the **answers** environment, an optional parameter [knuth] appears. The value of this parameter is a unique name for the solution to the quiz. Notice also, the **solution** environment follows, and is not nested within the **answers** environment.

7.4. How to Modify the quiz Environment

There are four ways the appearance of the quizzes can change:

- change the titles
- change the 'check' appearance
- change the text field in which the score appears,
- change the appearance of the 'Correction' button.

This section discusses each of these four in turn.

- **The Quiz Titles**

It is possible to redefine the quiz titles and other labels if desired.

▶ Locally:

```
\renewcommand\bqlabel{Begin Exam}
\renewcommand\eqlabel{End Exam}
```

- Globally:

```
\makeatletter
\renewcommand\eq@bqlabel{Begin Exam}
\renewcommand\eq@eqlabel{End Exam}
\makeatother
```

- **The check appearance**

The appearance of the 'check' can be chosen using the `\symbolchoice` macro of the exerquiz package. The permissible values for the argument of `\symbolchoice` are `check` (the default), `circle`, `cross`, `diamond`, `square`, and `star`.

This quiz was generated by inserting `\symbolchoice{diamond}` before the quiz.

Begin Quiz Answer each of the following. Passing is 100%.

1. Who created TEX?

☐ Knuth ☐ Lamport ☐ Carlisle ☐ Rahtz

2. Who originally wrote LaTeX?

☐ Knuth ☐ Lamport ☐ Carlisle ☐ Rahtz

End Quiz Score: Correct

- **Change color of Correction Marks**

The colors used to mark the quiz can be changed. Below are the defaults.

```
\renewcommand\checkColor{color.red}
\renewcommand\crossColor{color.red}
\renewcommand\correctColor{["RGB", 0, .6, 0]} % webgreen
```

- **The 'Correction' Button**

The 'Correction' button is defined by the `\eqButton`, which takes one argument; namely, the field name that contains the total score for the quiz, see the above examples. It also has one optional argument that can be used to modify the appearance of the button.

Local	Global	Default	Description
\BC	\eq@BC	1 0 0	border color
\BG	\eq@BG	.7529 .7529 .7529	face color
\CA	\eq@CA	Correct	button text
\RC	\eq@RC	My Answers!	rollover
\AC	\eq@AC	Please!	pushed text
\DA	\eq@DA	/Helv 10 Tf 0 g	text format
\BS	\eq@BS	/W 1 /S /B	button spec

Table 2: \eqButton Parameters

The macros listed in the first column of Table 2 are permitted in the optional parameter field of \eqButton—there is not parameter checking, TEX or Distiller/Reader will find the errors.

The meaning of these values is beyond the scope of this manual. Refer to the internet article "Pdfmarks: Links and Forms", [9], for details; in particular, see the "Forms" article.

Begin Quiz Answer each of the following. Passing is 100%.

1. What TEX System does Thomas Esser maintain?

☐ MikTEX ☐ csTEX ☐ teTEX ☐ fpTEX

2. What TEX System does Fabrice Popineau maintain?

☐ MikTEX ☐ csTEX ☐ teTEX ☐ fpTEX

3. What TEX System does Christian Schenk maintain?

☐ MikTEX ☐ csTEX ☐ teTEX ☐ fpTEX

End Quiz Score: TeX

The new part is the customized 'Correction' button. Here is a verbatim listing of the `\ScoreField` and `\eqButton` macros.

```
\ScoreField{qz:TeX-c}%
   \eqButton[\BC{0 0 1}      % blue border color
   \CA{TeX}                  % Button text
   \RC{Users}                % rollover text
   \AC{Group}                % pushed text
   \DA{/TiRo 10 Tf 0 0 1 rg}% times roman, 10 pt, blue text
   \BS{/W 1 /S /I}           % border width 1, inset button
```

```
]{qz:TeX-c}
```

▶ Thanks to Dan Luecking, these optional arguments are not sensitive to spaces between them, thus,

```
\eqButton
[
    \BC{0 0 1}  \BC{0 0 1}
    \CA{TeX}\RC{Users}
    \AC{Group} \DA{/TiRo 10 0 0 1 rg}
    \BS{/W 1 /S /I}
]{qz:TeX}
```

works as well.

▶ This example—as well as others—appears in webeqtst.tex, a test file that accompanies the AcroTEX Bundle.

• The Score Field

The score field is the text field to which the quiz (and its underlying JavaScript) report the score. This field can be constructed using the \ScoreField macro; e.g.,

(\ScoreField{qz:TeX-c})

We have seen many examples of the use of this macro.

In the simplest case, \ScoreField takes one argument, as above, the quizfieldname of the associated quiz. It's expansion produces a read-only text field that is 1.5 inches in width with a red border. The initial text that appears in the field is the expansion of the macro \eqScore. The expansion of \eqScore depends on the language option: \eqScore expands to 'Score:' by default, to 'Punkte:' for the german option and to 'Score :' for the french option.

The macro \ScoreField also has an optional parameter that can be used to modify the appearance of the text field. Should want to change the basic look of the text field produced by \ScoreField, just introduce the changes through this optional parameter. For example, the field

▶ See the file qz02.tex for details and examples of how to modify the quiz titles. The language files, e.g., eqfr.def and eqde.def, demonstrate how to redefine all variables, including those listed above.

8. Objective Style Questions

Beginning with version 2.0 of exerquiz, objective style questions can be posed. Single questions can be posed in the `oQuestion` environment, multiple questions can be placed in either the `shortquiz` or the `quiz` environments. This section discusses this type of question and all of its supporting commands.

8.1. Math and Text Questions

Exerquiz distinguishes between two types of open ended or objective questions:

1. A mathematical question that requires a mathematical expression as the answer.
2. A question that requires a text answer.

▶ The demo file jquiztst.tex is an important source of examples and instruction for the mathematical type question; the file jtxttst.tex has many examples for the text type question.

- **The Mathematical Question**

At this stage in the development of exerquiz, a (mathematical) question can be posed that requires an answer that is a function of one or more declared variables x, y, z, etc. Thus, when the declared variables x, y, z are given a value, the answer is reduced to a number.

For example, the answer to the question "Differentiate $\frac{d}{dx}\sin^2(x)$", is a function in one variable x, it can be evaluated numerically and can, therefore, be posed:

▶ Differentiate $\dfrac{d}{dx}\sin^2(x) =$ ☐

See '`\RespBoxMath`: The Math Question' on page 126 for details.

In contrast, consider the question: "Name the probability distribution popularly referred to as the 'bell-shaped curve' ". The answer to this question cannot be reduced to a numerical value. This question can be posed as an text objective question, or, it does lend itself to a multiple choice question, however.

- **The Text Question**

You can also pose question that require a text answer; for example,

▶ Name the probability distribution popularly referred to as the "bell-shaped curve". [ ]

See '\RespBoxTxt: The Text Question' on page 132 for details.

8.2. The oQuestion Environment

The **oQuestion** environment is a very simple environment for posing a *single* question and will be used in this section to discuss in detail the macros for posing mathematical and text open questions.

The syntax for the **oQuestion** environment is

```
\begin{oQuestion}{<field_name>}
<A math or text open ended question.>
\end{oQuestion}
```

The environment takes one required argument, a unique name for the question. This name, **field_name**, is used by other supporting macros.

- **`\RespBoxMath`: The Math Question**

The `\RespBoxMath` command is used for posing an objective question. This command must appear in the `oQuestion`, `shortquiz` or `quiz` environments. In this section we discuss only the `oQuestion` environment.

The following is a minimal example, additional enhancements will be discussed in subsequent sections.

▶ Differentiate $\dfrac{d}{dx}\sin^2(x) =$ [ ]

The code for the above example is

```
\begin{oQuestion}{sine1}
\redpoint Differentiate $\dfrac d{dx} \sin^2(x) =
\RespBoxMath{2*sin(x)*cos(x)}{4}{.0001}{[0,1]}$
\end{oQuestion}
```

The `\RespBoxMath` need not appear in math mode.

You can also pose multivariate questions as well, for example

▶ $\dfrac{\partial}{\partial y}4x^2y^3 =$ [ ]

The code for the above example is

```
\begin{oQuestion}{multivariate}
\redpoint $\dfrac{\partial}{\partial y} {4 x^2 y^3 }
    = \RespBoxMath{12*x^2*y^2}(xy){4}{.0001}{[0,1]x[0,1]}$
\end{oQuestion}
```

See the file `multivar.tex` for more examples quizzes involving multivariate problems.

The algorithm used for determining the correctness of the answer entered by the user is very simple: The user's answer and the correct answer are evaluated at randomly selected points in an interval, then compared. If any of the comparisons differ by more than a preselected amount, an ϵ value, if you will, the user's answer is declared incorrect; otherwise, it is considered correct.[4]

The command `\RespBoxMath` take ten parameters, five optional and five required:

```
\RespBoxMath[#1]#2(#3)[#4]#5#6#7#8[#9]*#10
```

[4]The idea for evaluating user input in this way comes from Drs. Wlodzimierz Bryc and Stephan Pelikan of The University of Cincinnati.

Parameters:

`#1` : Optional parameter used to modify the appearance of the text field. See The 'Correction' Button for examples, and exerquiz.dtx for a listing of all controlling macros.

`#2` : The correct answer to the question. This must be a numerical value, or a function of one variable. JavaScript Note: In JavaScript, functions such as `sin(x)` and `cos(x)` are methods of the `Math` object. It is not necessary, however, to type `Math.sin(x)` or `Math.cos(x)`; this is done by inserting the expression into a `with(Math)` group. For example,

```
with(Math){ 2*sin(x)*cos(x) }.
```

`#3` : An optional parameter, *delimited by parentheses*, that defines the independent variable; `x`, is the default value. Note that this parameter is set off by parentheses. For a multivariate question, just list the variables in juxtaposition, `(xyz)`. See the example in 'Some Enhancements' on page 135 section below.

`#4` : Optional, a named destination to the solution to the question. If

this parameter appears, then a solution must follow the question, enclosed in a `solution` environment.

`#5` : The number of samples points to be used, usually 3 or 4 is sufficient.

`#6` : Precision required, the ϵ value, if you will.

`#7` : Parameters #7 and #8 are used to define the interval from which to draw the sample points. There are two forms: (1) #7 is the left-hand endpoint of the interval and #8 is the right-hand endpoint (the use of #7 and #8 in this form is deprecated); (2) the interval is defined by standard interval notation, `[a,b]`. For a multivariate question—one where parameter #2 lists more than one variable, separate the intervals for each variable by a 'x', `[0,2]x[1,2]x[3,4]`.

`#8` : (1) #8 is the right-hand endpoint of the interval (the use of this parameter is deprecated); (2) in the second case, #8 is not used.

`#9` : This optional parameter is the name of a customized comparison function.

`#10:` (Only detected if following an asterisk, '`*`') The name of a JavaScript function that is to be used to process the user input.

▶ For the above example,

```
\RespBoxMath{2*sin(x)*cos(x)}{4}{.0001}{[0,1]}
```

no optional parameter is specified; the correct answer written in valid JavaScript is `2*sin(x)*cos(x)`; evaluation of the user's answer is done by randomly selecting 4 points from the interval $[0, 1]$; if the evaluation at any of the 4 points differs from the evaluation of the correct answer at the same point by more than $\epsilon = 0.0001$, the user's answer is considered wrong.

Once you choose the question to ask, you must then select the values of the parameters for `\RespBoxMath`.

▶ `Some Comments`:

1. The correct answer can be written either with valid JavaScript, or in the same syntax a user would enter the answer with. The functions and operators are pretty much as expected. See the the

demo file jquiztst.tex for some discussion how authors and users should enter their answers.

2. The interval from which the sample points are taken needs to be chosen with care. The interval must, obviously, be a subset of the domain of the answer function. Choose an interval away from any singularities the answer may have.

3. The JavaScript of Acrobat 5.0 does have exception handling, but this has not been incorporated into the code yet. Taking advantage of this new capability will be my next project. Exception handling will give the code protection against user's entering spurious answers. For example, based on the correct answer, the author chooses the interval $[0, 1]$, but the user enters a function whose domain does not contain the interval, such as `(x-1)^(1/2)`.

▶ See the file jquiztst.pdf for various examples of the math questions. The source code is available from the main Web/Exerquiz Web Site

• `\RespBoxTxt`: The Text Question

You can also pose question that takes a simple text response. The basic command for posing this type of question is `\RespBoxTxt`. Consider the example given earlier:

▶ Name the probability distribution popularly referred to as the "bell-shaped curve". [ ]

The underlying JavaScript compares the user's response against acceptable alternatives, as supplied by the author of the question. If there is a match, the response is deemed correct.

The code for this example is

```
\begin{oQuestion}{exTxt1}
\redpoint Name the probability distribution popularly
referred to as the ‘‘bell-shaped curve’’.\
\RespBoxTxt{0}{0}{4}{Normal}{Normal Distribution}%
{Gaussian}{Gaussian Distribution}
\end{oQuestion}
```

The command `\RespBoxTxt` takes five or more parameters.

```
\RespBoxTxt[#1]#2#3[#4]#5<plus listing of alternatives>
```

Parameters:

`#1` : Optional parameter used to modify the appearance of the text field. See The 'Correction' Button for examples, and exerquiz.dtx for a listing of all controlling macros.

`#2` : This required parameter is a number that indicates the filtering method to be used. Permissible values of this parameter are

`-1`: (The default) The author's and user's answers are not filtered in any way. (Spaces, case, and punctuation are preserved.)

`0`: The author's and user's answers are converted to lower case, any white space and non-word characters are removed.

`1`: The author's and user's answers are converted to lower case, any white space is removed.

`2`: The author's and user's answers are stripped of any white space.

See the JavaScript function `eqFilter` in `exerquiz.dtx` for the program code details. Additional filtering options may be added.

`#3` : This parameter a number that indicates the compare method to be used. Permissible values of this parameter are

- `0:` (The default) The author's and user's answers are compared for an exact match. (These answers are filtered before they are compared.)
- `1:` The user's response is searched in an attempt to get a substring match with the author's alternatives. Additional comparison methods may be added.

See the JavaScript function `compareTxt` in `exerquiz.dtx` for the program code details.

`#4` : Optional, a named destination to the solution to the question. If this parameter appears, then a solution must follow the question, enclosed in a `solution` environment.

`#5` : This required parameter is the number of alternative answers that are acceptable. The alternative answers are listed immediately after this parameter. (The example above specified that 4 alternatives follow.)

▶ See the file jtxttst.pdf for examples of the differences between various combinations of filtering rules and comparison methods. The source code is available from the main Web/Exerquiz Web Site

8.3. Some Enhancements

There are several enhancements to the math (using `\RespBoxMath`) and text (using `\RespBoxTxt`) open-ended question beyond the minimal examples given earlier. These enhancements can be used within the `oQuestion`, the `shortquiz` and the `quiz` environments.

- **Including an Answer Key with `\CorrAnsButton`**

The correct solution can be included in the question as well; just include the command `\CorrAnsButton`. This command takes one parameter, the correct answer that will be viewed when the user clicks on the button.

The example below also illustrates the (optional) third parameter of `\RespBoxMath`. Here we pose the question in the variable t rather than the default variable of x.

▶ Differentiate

$\frac{d}{dt}\sin^2(t) =$ [ ] Ans

The listing follows:

```
\begin{oQuestion}{sine2}\\[1ex]
\redpoint Differentiate $\dfrac d{dt} \sin^2(t) =$
\RespBoxMath{2*sin(t)*cos(t)}(t){4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*sin(t)*cos(t)}
\end{oQuestion}
```

The `\CorrAnsButton` takes one parameter, the correct answer. This answer is (usually) the same as the one given as the second argument (the optional argument is the first) in the `\RespBoxMath` command.

▶ The `\CorrAnsButton` also controls access to the (optional) solution, see the next section.

• Including a Solution

In addition to a correct answer, you can also include a solution to the question. Insert the optional fourth parameter—fourth for both

`\RespBoxMath` and `\RespBoxTxt`—into the parameter list giving the name of the destination to the solution. Follow the question by a `solution` environment containing the solution.

The user `Shift-Clicks` on the `\CorrAnsButton` to jump to the solution.

▶ Differentiate

$\frac{d}{dt}\sin^2(t) =$ [ ] Ans

The listing follows:

```
\begin{oQuestion}{sine3}\\[1ex]
\redpoint Differentiate $\dfrac d{dt} \sin^2(t) =$
\RespBoxMath{2*sin(t)*cos(t)}(t)[sine3]{4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*sin(t)*cos(t)}
\begin{solution}
$$
    \frac d{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)
$$
\end{solution}
\end{oQuestion}
```

▶ The `\CorrAnsButton` works the same way for the `shortquiz` and the `quiz` environments.

- **Including a Tally Box**

The macro `\sqTallyBox` is used to keep a running total of the number of wrong answers a user has entered into the response box.

For example,

▶ Differentiate

$\frac{d}{dx}\sin^2(x) =$ [ ] Ans [ ]

The listing follows:

```
\begin{oQuestion}{sine4}
\redpoint Differentiate\\[1ex]
$\dfrac d{dx} \sin^2(x) =$
\RespBoxMath{2*sin(x)*cos(x)}{4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*sin(x)*cos(x)}\kern1bp
\sqTallyBox
\end{oQuestion}
```

▶ The tally box can be used within the `oQuestion` and `shortquiz` environments; in the `quiz` environment, no tally box is used.

- **Clearing the Fields**

For the `oQuestion` and the `shortquiz` environments, you can clear the response box fields by placing insert `\sqClearButton`.

▶ Differentiate

$\frac{d}{dx}\sin^2(x) =$ [ ] Ans [ ] Clear

The listing follows:

```
\begin{oQuestion}{sine5}
\redpoint Differentiate\\[1ex]
$\dfrac d{dx} \sin^2(x) =$
\RespBoxMath{2*sin(x)*cos(x)}{4}{.0001}{0}{1}%
\CorrAnsButton{2*sin(x)*cos(x)}\kern1bp
\sqTallyBox\kern1bp\sqClearButton
\end{oQuestion}
```

You'll notice that I've inserted a `\kern1bp` to separate the two fields `\sqTallyBox` and `\sqClearButton`, this is to keep their borders

from overlapping.

8.4. The `shortquiz` Environment

The objective question (with or without the presence of a correction box, `\CorrAnsButton` or a tally box `\sqTallyBox`) can be mixed in with multiple choice questions.

Solutions to the questions can also be included using a `solution` environment. Click on the "Ans" button to get the answer to a question; shift-click on the "Ans" button to get the solution.

Quiz Answer each of the following. Passing is 100%.

1. If f is differentiable, then f is continuous.

(a) True (b) False

2. $\dfrac{d}{dx}\sin^2(x) =$ Ans

3. Name *one* of the two people recognized as a founder of Calculus.

Ans

Clear

▶ When using objective questions within a **shortquiz** environment, you must give a unique field name as an optional argument of the environment. The listing of this example follows:

```
\begin{shortquiz}[oQsq]  % <-- unique field name
Answer each of the following. Passing is 100\%.
\begin{questions}

\item If $f$ is differentiable, then $f$ is continuous.
\begin{answers}{4}
\Ans1 True & \Ans0 False
\end{answers}\hfill\sqTallyBox

\item $\displaystyle\frac d{dx} \sin^2(x) =$
\RespBoxMath{2*sin(x)*cos(x)}[sinsqx]{4}{.0001}{0}{1}%
\hfill\CorrAnsButton{2*sin(x)*cos(x)}%
\kern1bp\sqTallyBox
\begin{solution}
$$
    \frac d{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)
$$
\end{solution}
```

```
\item Name \emph{one} of the two people recognized
as a founder of Calculus.\vadjust{\kern3pt}\newline
\RespBoxTxt{2}{0}[newton]{5}{Isaac Newton}{Newton}{I. Newton}%
{Gottfried Leibniz}{Leibniz}\hfill
\CorrAnsButton{Isaac Newton or Gottfried Leibniz}%
\kern1bp\sqTallyBox
\begin{solution}
Yes, Isaac Newton and Gottfried Leibniz are considered
founders of Calculus.
\end{solution}
\end{questions}
\end{shortquiz}
\begin{flushright}
\sqClearButton\kern1bp\sqTallyTotal %<-- \sqTotal=total tally
\end{flushright}
```

Example Notes:

- Note the optional argument, giving this collection of questions a common base name. All supporting macros use this name.
- The named destination to the solution is entered with parameter #5 of \RespBoxMath, and with parameter #4 of \RespBoxTxt.

- In this example, another built-in macro, `\sqTallyTotal` was used. This macro creates a text field that accumulates the totals of all the tally boxes.

▶ The `shortquiz` environment can also be used for a single objective question. Just don't use the `questions` environment within.

```
\begin{shortquiz}[anExample]
< an objective style question >
\end{shortquiz}
```

8.5. The `quiz` Environment

Objective questions can be mixed in with multiple choice question within the `quiz` environment. When posing an objective style question in the `quiz` environment, use the `\RespBoxMath` and `\RespBoxTxt` commands and optionally include the `\CorrAnsButton`.

Since the evaluation of the quiz is delayed until the user has finished the quiz, the `\sqTallyBox` macro is not needed.

Begin Quiz Answer each of the following. Passing is 100%.

1. If f is differentiable, then f is continuous.

☐ True ☐ False

2. $\frac{d}{dx}\sin^2(x) =$ _____

3. Name *one* of the two people recognized as a founder of Calculus.

_____

End Quiz Score: Correct

Answers: _____

▶ The buttons created by `\CorrAnsButton` are hidden until the user ends the quiz (and gets scored) and clicks on the corrections button (`\eqButton`). The `\CorrAnsButton` should not be included if there is no `\eqButton`.

▶ If there is a solution to the problem, the "Ans" button is outlined in green. Shilf-click on the "Ans" button to jump to the solution.

▶ The `quiz` environment requires a field name, this same name is used by the objective style question as well.

The listing for the above example follows.

```
\begin{quiz}*{oQq}
Answer each of the following. Passing is 100\%.
\begin{questions}

\item If $f$ is differentiable, then $f$ is continuous.
\begin{answers}{4}
\Ans1 True & \Ans0 False
\end{answers}

\item $\displaystyle\frac d{dx} \sin^2(x) =$
\RespBoxMath{2*sin(x)*cos(x)}{4}{.0001}{0}{1}%
\hfill\CorrAnsButton{2*sin(x)*cos(x)}%

\item Name \emph{one} of the two people recognized
as a founder of Calculus.\vadjust{\kern3pt}\newline
\RespBoxTxt{2}{0}[leibniz]{5}{Isaac Newton}{Newton}{I. Newton}%
{Gottfried Leibniz}{Leibniz}\hfill
\CorrAnsButton{Isaac Newton or Gottfried Leibniz}
\begin{solution}
Yes, Isaac Newton and Gottfried Leibniz are considered
founders of Calculus.
\end{solution}
```

```
\end{questions}
\end{quiz}\quad\ScoreField{oQq}\eqButton{oQq}

\noindent Answers: \AnswerField{oQq}
```

8.6. Extending AcroTEX with `dljslib` and `insdljs`

The exerquiz Package, especially the math fill-in question, is quite programmable. In this section, we discuss to methods of extending the capabilities of the AcroTEX Bundle: (1) through the use of the package `dljslib`, which is a JavaScript library of extensions; (2) by writing your own custom extensions using the `insdljs` package for inserting JavaScripts into the PDF document.

- **Using the `dljslib` Package**

The `dljslib` Package is actually a "library" of JavaScript functions. At the time of this writing, the library has JavaScripts that can process process answers to math fill-in questions where an *equation* or a *vector* answer is expected. There is also a JavaScript compare function that properly evaluates an answer when an indefinite integral is

expected. See the documentation that accompanies the package (by latexing `dljslib.dtx`) for details of how to use the library.

▶ **Equation handling.** See the sample file `jqzequat.tex` for examples of posing and evaluating questions that expect an equation as the response. Below is a portion of the preamble of that file; basically, to use one or more of the JavaScript in the JavaScript library, you specify that option `\usepackage` command for `dljslib` pacakge. In this case, we want to process equations so type...

```
\documentclass{article}
\usepackage{amsmath,amscd}
\usepackage[tight,pdftex,designi,nodirectory]{web}
\usepackage{exerquiz}
\usepackage[equations]{dljslib}  % <--choose equations
```

▶ **Vector Handling** There are also JavaScript functions for processing vector answers. See the sample file `jqzspec.tex`. Actually this file does not use the JavaScript library, but is more of a tutorial on how to use `\insdljs` to write custom JavaScripts to process exerquiz math fill-in questions.

The preamble of that document could actually be replaced with. . .

```
\documentclass{article}
\usepackage{amsmath,amscd}
\usepackage[tight,dvipdfm,designi,nodirectory]{web}
\usepackage{exerquiz}
\usepackage[vectors,indefIntegral]{dljslib}
```

The `vectors` options specifies JavaScripts for processing vector questions. The `indefIntegral` option is also specified. This is because that in the file `jqzspec.tex` a comparison function is developed for properly evaluating questions in which an indefinite integral is expected.

▶ In addition to the two above mentioned sets of JavaScripts there are a couple of comparison functions, one for processing indefinite integrals (see `dljs_ex.tex`), and the other for using relative absolute error rather than absolute error. Again, see the documentation of `dljslib.dtx` and the sample file `jqzspec.tex`.

- **Using the insdljs Package**

With the **insdljs** Package you can write your own JavaScript functions right in the LaTeX source file. These custom JavaScripts are then inserted into the section of the PDF document where the document-level Javascripts reside. This package is a stand-alone package, and does not need exerquiz, though exerquiz now uses this package to insert its JavaScripts into the document.

See the documentation that accompanies the package (by latexing **insdljs.dtx**) for details of how to use the library. Also, see the sample file **insdljs_ex.tex** for a examples that do not use exerquiz, and the file **jqzspec.tex**, for examples that do use exerquiz.

9. Submitting a quiz to a Web Server

Quizzes created by the **quiz** environment are entirely self-contained. They function within the Web browser (or from within the Acrobat Reader) and do not communicate with any server. This kind of quiz is ideal for a do-it-yourself tutorial system, read by a well-motivated student who has the discipline to read the material and to take the

quizzes in the spirit in which they are given.

However, some educators, myself included, may wish to use the quizzes created by the `quiz` environment for classroom credit. It is necessary, therefore, for the student to be able to submit quiz results to a Web server which, in turn, should store the results to a database.

In this section we discuss techniques of turning the quiz into something that can be submitted to a server.

9.1. Technical Info for "Do It Yourself"

All one really has to do is to redefine the "End Quiz" link or button to submit the results of the quiz to the Web server and CGI of your choice. Since the quiz itself is scored, (optionally) marked, with (optional) answers and solutions provided, the CGI simply stores the quiz results to a database.

- **Redefining "End Quiz"**

I've written the "End Quiz" link (button) to have various programming hooks available to the developer.

The following code is common to both \eq@EndQuizLink and \eq@EndQuizButton, the macros that control the action of the end quiz link and button, respectively.

```
if (\minQuizResp(\thequestionno)) {\jsR\jsT
    var f = this.getField("ScoreField.\curr@quiz");\jsR\jsT\jsT
    if ( f != null )\jsR\jsT\jsT\jsT
        this.getField("ScoreField.\curr@quiz").value
            =(\eq@QuizTotalMsg);\jsR\jsT\jsT
    \eq@submitURL
    resetQuiz("\curr@quiz")\jsR\jsT
}
```

▶ The code is a mixture of LaTeX macros and JavaScript. You can see from this code, that there is a submit hook macro provided, \eq@submitURL. Normally, this macro has a definition of \empty. A developer need only redefine this macro accordingly; one would use the Acrobat JavaScript method this.submitForm() to do this. See the *Acrobat JavaScript Object Specification* [1] for more detail about this method.

▶ The code flow above is as follows: (1) Execute this code if the

threshold has been met. (See Setting the Threshold.) The text macro \curr@quiz holds the base name of the current quiz.

(2) If the field "ScoreField.\curr@quiz" exists, then write the student's score to that field (This is the "Score: 2 out of 3" that you see in the demo quizzes.)

(3) We then submit with the macro \eq@submitURL. (This would do nothing if its value is \empty, the default value.) At this point we call a DLJS resetQuiz("\curr@quiz") which sets some values in an array to indicate the state of this quiz.

• Gathering ID Information with \eqTextField

▶ What kind of information would one submit to a CGI? Well, there is the usual information concerning the identity of the student (Name, SSN, etc.) and the course, section and so on.

This basic information can be gathered from the student by inserting text fields into the document to be filled in. Exerquiz provides the macro \eqTextField[5] for this purpose. For example,

[5] You can also use hyperref's \TextField command for this purpose as well.

```
\newcommand\FirstName[2]{\eqTextField
    [\DV{First Name}\DA{/TiRo 10 Tf 0 0 1 rg }]
    {IdInfo.Name.First}{#1}{#2}}
```

This defines a text field with a name of "IdInfo.Name.First", the two arguments are the width and height of the field that you want to create. E.g.,

```
\FirstName{100pt}{10pt}
```

creates a text field 100pt wide and 10pt high.

The \eqTextField macro takes four parameters.

```
\eqTextField[#1]#2#3#4
```

The first (optional) parameter can be used to custom design the field; the second is the name of the field; the third and fourth are the width and height of the field desired.

- **Gathering Quiz Specific Information with \eqSubmit**

In addition to ID information on the one taking the quiz, specific information about what quiz is being taken and where the results of the quiz are to be stored are needed as well.

Exerquiz provides a basic macro, called \eqSubmit that can be used to gather basic formation of this type. The definition of it and related commands are given below:

```
\newcommand\databaseName[1]{\def\db@Name{#1}}\def\db@Name{}
\newcommand\tableName[1]{\def\db@Table{#1}}\def\db@Table{}
\newcommand\eqCGI[1]{\def\eq@CGI{#1}}\def\eq@CGI{}
\newcommand\eqSubmit[3]
    {\eqCGI{"#1"}\databaseName{#2}\tableName{#3}}
```

The meaning of the parameters are self-explanatory.

Just prior to the quiz you can type:

```
\eqSubmit{http://www.myschool.edu/cgi-bin/myCGI.cgi}%
    {CalcIII}{Quizzes}
\begin{quiz}*{Quiz3} Answer each of the following.
\begin{questions}
...
...
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz

\noindent
Answers: \AnswerField\currQuiz
```

▶ Any redefinition of `\eq@submitURL` would then include the values of some or all of these text parameters:

`\eq@CGI, \db@Name, \db@Table, \curr@quiz`

The last text macro is not gatherd by `\eqSubmit`, but is certainly known at the time `\eq@submitURL` is expanded.

• Some Variables to Submit

When you submit a quiz to a server, the values of *all* fields are also submitted, unless you define specifically which fields are to be submitted.

In addition to the ID info, you would like also to submit the results of the quiz itself. The relevant variables are as follows:

1. The JavaScript variable `Score` has the number of correct responses as its value.
2. The LaTeX counter variable `\thequestionno` has the count of the total number of questions in the quiz.
3. The JavaScript array `Responses` contains the responses of the student: multiple choice and fill-in responses. The contents of this

array can be converted to a comma-delimited string by using the `toString()` method, `Responses.toString()`.

Now, how does one submit these values? The `\eq@submitURL` command can be used not only to submit the data, but to also populate certain *hidden* fields with this information. The hidden data is submitted along with the ID info to be processed. You can use the `\eqTextField` to create hidden text fields for this purpose. See the next section for a discussion of how to create hidden text fields.

9.2. The eq2db Package

Currently, I am working on a package, which I call `eq2db`, designed to make the tasks, as outlined in the Section 9.1, easy and routine. As the name suggests, this package facilitates submitting an `Exerquiz` quiz to a CGI for storage in a database.

The package itself does very little other than to define some useful commands, such as

```
\newcommand\hiddenTextField[3][]{\eqTextField
    [\DV{#3}\V{#3}\F2#1]{#2}{10bp}{10bp}}
```

which can be used for creating hidden text fields. These hidden text fields can then be populated at submission time by the values of the quiz: `Score`, `Responses.toString()`, `\thequestionno`.

The `eq2db` currently has only one option, `eqRecord`:

```
\usepackage[eqRecord]{eq2db}
```

The option `eqRecord` sets up the quiz to use an ASP (Active Server Page) that I have written. This ASP, named naturally, `eqRecord.asp`, takes the data and stores it to a database, such as Microsoft Access.

There will also be a `custom` option. With this option, a developer can write LaTeX code to set the quiz up for submission to a CGI used or written by the developer.

More details and demos of this package when the package is released, hopefully, by the first quarter of 2002.

9.3. Features *apropos* to Submitting

- **Assigning Points**

The questions on a quiz, especially a quiz meant for credit, may not have the same weight. A point scheme, therefore, has been created;

several additional text fields in support have also been defined.

Here is a simple two question example to illustrate:

Begin Quiz Answer each of the following. Passing is 100%.

1. (4^{pts}) If $\lim_{x \to a} f(x) = f(a)$, then we say that f is...

☐ differentiable ☐ continuous ☐ integrable

2. (6^{pts}) Name *one* of the two people recognized as a founder of Calculus.

End Quiz Score: Correct

Answers:

Points: Percent:

▶ See the sample file quizpts.tex for a more elaborate version of this question, as well as the source code.

1. \PTs#1: This macro takes one argument, the number of points to be assigned to the current problem. Place this command immediately after the \item in the questions environment. For example, in the above quiz we had

```
\item\PTs{6} Name \emph{one} of the two people recognized
             as a founder of Calculus.
```

2. \PTsHook#1: This macro, which takes on argument, can be used to type set the points assigned. and is called by \PTs. The argument is what is to be typeset. The value assigned the current problem by \PTs is contained within the macro \eqPTs. In the quiz above, we had

```
\PTsHook{($\eqPTs^{\text{pts}}$)}
```

3. There are three other commands that create text fields to display results from a quiz with points assigned:
   - \PointsField[#1]#2: The number of points earned for the quiz, the total points are also reported. The parameter #2 is the base name of the quiz.

- \PercentField[#1]#2: The percentage score for the quiz. The parameter #1 is the base name of the quiz.
- \GradeField[#1]#2: The letter grade of the performance on the quiz. The parameter #2 is the base name of the quiz. The values placed in this field are determined by the macro \eqGradeScale.

4. \eqGradeScale: This macro sets the grade scale of a quiz, the default definition is

```
\newcommand\eqGradeScale{"A",[90, 100],"B",[80,90],
    "C",[70,80],"D",[60,70],"F",[0,60]}
```

The ways things are defined now, there can be only one grade scale per document. The value of \eqGradeScale is a matrix with an even number of elements. The odd numbered elements are the grades; the even number elements are intervals of percentages (percentages of the total number of points on the quiz). If the percentage of the score falls into a particular range, the corresponding grade is assigned.

Note, obviously, you can redefine this command. The letter grades

do not actually have to be grades, they can be little messages to the student upon completion of the quiz.

```
\renewcommand\eqGradeScale{%
    "Excellent Work.",[90, 100],
    "Solid Effort.",[80,90],
    "Fair.",[70,80],
    "Needs improvement, better work expected.",[60,70],
    "Learning still in progress.",[0,60]
}
```

- `\NoPeeking`

If you execute the command `\NoPeeking` in the preamble of your document, or prior to a `quiz`, then any quiz question with solution will be protected somewhat from prying eyes.

In this case, an open page action is placed on the first page of each solution. If the user (student) tries to view a quiz solution before doing the quiz, the Acrobat Reader will automatically change the page to the page containing the quiz and place an alert box on the screen saying that viewing the solution before taking the quiz is not permitted.

To resort to the default behavior, use the `\AllowPeeking` command.

The previous `quiz` has been surrounded with a `\NoPeeking`/-`\AllowPeeking` pair. If you go to one of the solutions to that quiz, you will see what happens. If nothing interesting happens, read the next red point.

▶ Protection is removed when you click on "End Quiz" and restored when you click on some "Begin Quiz".

10. List of Options

Options of the Web/Exerquiz Packages	
Options of the Web Package	
`dvipsone`	dvi-to-ps driver by Y&Y, Inc.
`dvips`	dvi-to-ps driver
`pdftex`	tex-to-pdf application
`dviwindo`	Y&Y's dvi previewer (links work in previewer)
`dvipdfm`	dvi-to-pdf application
`textures`	the Textures System for Mac
`designi`, `designii`, `designiii`	these set screen design parameters
`navibar`	inserts a menu bar at the bottom or each page

Options of the Web/Exerquiz Packages (cont.)	
`latextoc`	displays the standard toc
`nodirectory`	eliminates the directory listing on the title page
`forpaper`	this turns off color, and does not put solutions on separate pages.
`latexlayout`	`web` uses page layout for `article` class. For use with `forpaper`.
`tight`	redefines list environment parameters so lists don't take up so much space
`dutch`	Dutch for web, passed to exerquiz. (Thanks to Henny Wilbrink)
`french`	French for web, passed to exerquiz. (Thanks to Jean-Michel Sarlat)

Options of the Web/Exerquiz Packages (cont.)	
`german`	German for web, passed to exerquiz. (Thanks to Michael Wiedmann)
`italian`	Italian for web, passed to exerquiz. (Thanks to PierLuigi Zezza)
`norsk`	Norwegian for web, passed to exerquiz. (Thanks to Hans Fredrik Nordhaug)
`russian`	Russian for web, passed to exerquiz. (Thanks to Sergei V. Znamenskii)
`spanish`	Spanish for web, passed to exerquiz. (Thanks to Pedro Luis Luque)

Options of the Web/Exerquiz Packages (cont.)	
`polish`	Polish for web, passed to exerquiz. (Thanks to Jerzy Mycielski)
Options of the Exerquiz Package	
`pdftex`	tex-to-pdf application
`dviwindo`	Y&Y's dvi previewer (`exercise` environment only)
`dvipdfm`	dvi-to-pdf application
`nosolutions`	removes the solutions to the exercises
`nohiddensolutions`	overrides the 'h' (hidden) option for the exercises.
`noHiddensolutions`	overrides the 'h' and 'H' (hidden) options for the exercises.
`nocorrections`	removes the ability to correct the quizzes

Options of the Web/Exerquiz Packages (cont.)	
`solutionsafter`	solutions to exercises are typeset just after the question
`forpaper`	same function as in web. Needed when exerquiz is not used with web
`preview`	shows the outline of all form fields in the dvi previewer
`nodljs`	turns off the insertion of DLJS
`acrobativ`	equivalent to `nodljs`
`exercisesonly`	if document has only exercises, no doc level JS needed
`debug`	this option is passed on to the `insDLJS` package
`proofing`	mark the correct answers for `shortquiz` & `quiz` for proof reading.

Options of the Web/Exerquiz Packages (cont.)	
dutch	JavaScript messages in Dutch (Thanks to Henny Wilbrink)
french	JavaScript messages in French (Thanks to Jean-Michel Sarlat)
german	JavaScript messages in German (Thanks to Michael Wiedmann)
italian	JavaScript messages in Italian (Thanks to PierLuigi Zezza)
norsk	JavaScript messages in Noregian (Thanks to Hans Fredrik Nordhaug)
russian	JavaScript messages in Russian (Thanks to Sergei V. Znamenskii)
spanish	JavaScript messages in Spanish (Thanks to Pedro Luis Luque)

Options of the Web/Exerquiz Packages (cont.)	
`polish`	JavaScript messages in Spanish (Thanks to Jerzy Mycielski)

Solutions to Exercises

Exercise 1. We evaluate by `integration by parts`:

$$\begin{aligned}
\int x^2 e^{2x}\,dx &= \frac{1}{2}x^2e^{2x} - \int xe^{2x}\,dx && u = x^2,\ dv = e^{2x}\,dx \\
&= \frac{1}{2}x^2e^{2x} - \left[\frac{1}{2}xe^{2x} - \int \frac{1}{2}e^{2x}\,dx\right] && \text{integration by parts} \\
&= \frac{1}{2}x^2e^{2x} - \frac{1}{2}xe^{2x} + \frac{1}{2}\int e^{2x}\,dx && u = x^2,\ dv = e^{2x}\,dx \\
&= \frac{1}{2}x^2e^{2x} - \frac{1}{2}xe^{2x} + \frac{1}{4}e^{2x} && \text{integration by parts} \\
&= \frac{1}{4}(2x^2 - 2x + 1)e^{2x} && \text{simplify!}
\end{aligned}$$

Exercise 1

Exercise 2.

$$\boxed{x + y = 1}$$

Exercise 2

Exercise 3(a) Velocity is the rate of change of position with respect to time. In symbols:

$$v = \frac{ds}{dt}$$

For our problem, we have

$$v = \frac{ds}{dt} = \frac{d}{dt}(t^2 - 5t + 1) = 2t - 5.$$

The velocity at time t is given by $\boxed{v = 2t - 5}$. □

Exercise 3(b) Acceleration is the rate of change of velocity with respect to time. Thus,

$$a = \frac{dv}{dt}$$

For our problem, we have

$$a = \frac{dv}{dt} = \frac{d}{dt}(2t - 5) = 2.$$

The acceleration at time t is constant: $\boxed{a = 2}$. □

Exercise 4(a) $i^2 = -1$ □

Exercise 4(b) $i^3 = ii^2 = -i$ □

Exercise 4(c) $z + \bar{z} = \operatorname{Re} z$ □

Exercise 4(d) $\dfrac{1}{z} = \dfrac{1}{z}\dfrac{\bar{z}}{\bar{z}} = \dfrac{z}{z\bar{z}} = \dfrac{z}{|z|^2}$ □

Exercise 6(a) $v = 2t - 5$. □

Problem 8. This is the solution. ◀

Exercise 9. It is well known that $2 + 2 = 4$. Exercise 9

Project Hint: There, you didn't need my help after all. ◀

Solutions to Quizzes

Solution to Quiz: The answer is 'Yes'. The definition requires that

$$F'(x) = f(x) \quad \text{for all } x,$$

well, let's check it out.

The definition of f is $f(s) = 4s^3$ and so $f(x) = 4x^3$.

The definition of F is $F(t) = t^4$ and so, by the rules of differentiation, $F'(t) = 4t^3$. Thus, $F'(x) = 4x^3$. Therefore,

$$F'(x) = 4x^3 = f(x) \quad \text{for all } x,$$

as required by the definition. End Quiz

Solution to Quiz: If you erred on this one, more than likely it was on the appropriate multiplicative constant: 6 not 18. At least that's what I'm betting on.

The instructions of the LCD Algorithm said to *completely factor the denominator*. Here's a list of the factors

$$\underbrace{3, x^{3/2}, y^2}_{\text{first term}}, \underbrace{2, 3, x, y^4}_{\text{second term}}$$

Let's rearrange them

$$2, 3, 3, x, x^{3/2}, y^2, y^4$$

Now drop duplicate factors—that's the 3. Oops! I did mention dropping identical factors, didn't I?

$$2, 3, x, x^{3/2}, y^2, y^4$$

Now, group together all terms which have the same base, then drop, from each of these groups all terms but the one with the highest power. We obtain then,

$$2, 3, x^{3/2}, y^4$$

The LCD is the product of same:

$$\text{LCD} = (2)(3)x^{3/2}y^4 = 6x^{3/2}y^4.$$

Solution Notes: Alternative (a) will work as a common denominator, but it is not the least common denominator. If you use (a), you will be working with larger numbers than is really necessary. End Quiz

Solution to Quiz: Yes, Donald Knuth was the creator of TEX.

End Quiz

Solution to Quiz: Yes, Leslie Lamport was the creator of LaTeX.

End Quiz

Solution to Quiz:

$$\frac{d}{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)$$

End Quiz

Solution to Quiz:

$$\frac{d}{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)$$

End Quiz

Solution to Quiz: Yes, Isaac Newton and Gottfried Leibniz are considered founders of Calculus.

End Quiz

Solution to Quiz:

$$\frac{d}{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)$$

End Quiz

Solution to Quiz: Yes, Isaac Newton and Gottfried Leibniz are considered founders of Calculus.

End Quiz

References

[1] Acrobat JavaScript Object Specification, Version 5.0., #5186, Adobe Systems, Inc., 2001[6] 151

[2] Leslie Lamport, *LaTeX: A Document Preparation* System (2nd ed.), Addison-Wesley Publishing Company, 1994, ISBN 0-201-52983-1.

[3] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley Publishing Company, 1994, ISBN 0-201-54199-8. 77

[4] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley Publishing Company, 1997, ISBN 0-201-85469-4.

[5] Michel Goossens, and Rahtz, Sebastian, with Gurari, Eitan, Moore, Ross, and Sutor, Robert, *The LaTeX Web Companion*, Addison Wesley Longman, Reading, Massachusetts, USA, 1999. ISBN: 0-201-43311-7. 35

[6] http://partners.adobe.com/asn/developer/technotes/acrobatpdf.html

[6] Helmut Kopka and Patrick W. Daly, *A Guide to LaTeX2e* (2nd ed.), Addison-Wesley Publishing Company, 1995, ISBN 0-201-43777-X.

[7] Donald E. Knuth, *The TeXbook*, Addison-Wesley Publishing Company, 1987, ISBN 0-201-13448-9.

[8] Thomas Merz, *Web Publishing with Acrobat/PDF*, Springer-Verlag Berlin Heidelberg New York, 1998, ISBN 3-540-63762-1.

[9] D.P. Story, *Pdfmarks: Links and Forms*, AcroTeX Web Site, 1998, www.math.uakron.edu/~dpstory/acrotex.html 119

[10] D.P. Story, *Using LaTeX to Create Quality PDF Documents for the WWW*, AcroTeX Web Site, 1998, www.math.uakron.edu/~dpstory/acrotex.html 29