AcroTEX Software Development Team

@EASE

AcroTEX Exam Assembly System Environment

D. P. Story, Jürgen Gilg and Simon Singer

May 28, 2017

Version 1.0

Table of Contents

Preface **3**
- 1 Acrobat Requirements . . . 3
- 2 LaTeX Requirements . . . 4
  - 2.1 The atb Package . . . 4
  - 2.2 The atbdb Package . . . 4

1 The ATB Package **5**
- 1 Installation . . . 5
- 2 The atb Folder Structure . . . 6
- 3 Options of the atb Package . . . 8
- 4 Language Localizations . . . 9

2 The Control Panel for @EASE **10**
- 1 The Methodology . . . 10
- 2 Description of Controls . . . 11
  - 2.1 Class Options . . . 16
  - 2.2 eqexam Info . . . 17
  - 2.3 Preamble . . . 20
- 3 The @EASE Toolbar Buttons . . . 20
  - 3.1 Toolbar Buttons: Acrobat earlier than Version 10 . . . 20
  - 3.2 Toolbar Buttons: Acrobat 10 or later . . . 21
- 4 Building your First Exam . . . 22
  - 4.1 A Walk-through . . . 22
  - 4.2 Post Assembly Processing . . . 24

3 The @EASE Database **25**
- 1 The DB Document . . . 25
- 2 Passing preamble material to myExam.tex . . . 26
- 3 Instructions and Commenting . . . 26
- 4 Tagging the problems . . . 27
  - 4.1 Question Types and How to Tag Them . . . 28
    - Questions with No Parts, No Variations . . . 28
    - Questions with No Parts, with Variations . . . 29
    - Questions with Parts, No Variations . . . 29
    - Questions with Parts, with Variations . . . 29
    - Questions with Selectable Parts, with Variations . . . 31
    - Selecting Reference Charts, Tables, and Such . . . 31
  - 4.2 Multiple Choice with \eFreeze . . . 32
- 5 Additional Guidelines for a good DB Document . . . 34

4 The eqexam Package **37**
1 Introduction . . . 37
1.1 Required and Optional Packages . . . 37
2 Building an Exam . . . 38
2.1 The Preamble . . . 38
2.2 The exam Environment . . . 39
2.3 The problem and problem* Environments . . . 40
• problem . . . 41
• problem* . . . 42
• Page Breaking . . . 44
2.4 Special Constructs and other Gizmos . . . 45
2.5 Fill-in Questions . . . 45
• Short Fill-in Questions . . . 45
• True/False Questions . . . 45
• Long Fill-in Questions . . . 47
2.6 Multiple Choice . . . 47
2.7 Multiple Selection . . . 48
2.8 Gizmos and Gadgets . . . 49
• The workarea Environment . . . 49
• The \placeAtxy Command . . . 50
• The splitsolution Environment . . . 51
3 eqexam Options . . . 52
3.1 Configuration Files . . . 56
4 Bells, Whistles and other Customizations . . . 56
4.1 Customizations . . . 56
• Course Info Commands . . . 56
• Changing the Title and Cover Page . . . 57
• Changing the Running Headers . . . 59
• Localization of Strings . . . 60
4.2 Creating Multiple Versions of Exam . . . 61
4.3 The Point and Totals Boxes . . . 65
4.4 The eqComments Environment . . . 66
4.5 The \OnBackOfPage Command . . . 67
4.6 The \pushProblem and \popProblem Commands . . . 68

Preface

This manual contains the documentation for @EASE,[1] an exam assembly and database system for LaTeX users. This product is targeted at educators who use LaTeX to write their exams, quizzes and homework assignments.

@EASE allows the educator to build up, through time, a personal database of questions on a variety of topics of interest. With the @EASE control panel, the educator opens appropriate database files, visually selects questions of interest, then builds an exam consisting of the questions selected.

This manual covers the following topics:

- Chapter 1. The ATB Package: Discusses the installation of @EASE, and the underlying packages—atb and atbdb—used to create the @EASE control panel and @EASE database (DB) documents.
- Chapter 2. The Control Panel for @EASE: An overview of the **AcroTeX Exam Assembly System Environment** and a detailed discussion of the @EASE control panel.
- Chapter 3. The @EASE Database: Discusses how to develop and maintain your personal database.
- Chapter 4. The eqexam Package: @EASE uses the eqexam package. A subset of the documentation of eqexam is presented.

1. Acrobat Requirements

First things first: The major requirement of this @EASE is Acrobat Professional 7.0 or later;[2] to repeat

Acrobat Professional 7.0 or later and accompanying **Distiller**

are required for this package to perform as designed. The reason for this requirement is discussed in the section 'The Methodology' on page 10.

☛ **PDF Creator Application.** You can use Adobe Distiller (the preferred method[3]), pdflatex, or xelatex as your PDF creator application; in either case, Acrobat Pro 7.0 or later is still required. During the exam assembly phase, JavaScript API are used that are not available in Adobe Reader. Once the exam is assembled and made into a PDF (using Adobe Distiller, pdflatex, or xelatex), the PDF version of the exam can be viewed and printed in Adobe Reader.

[1] One of the members of the AcroTeX eDucation System Tools

[2] In the United States and Europe, Adobe offers a significant academic discount on its software, including Acrobat Pro 7.0. Educators should look into the price structure of Adobe Acrobat at their institutions; perhaps, their Department or College can supply a financial grant for the purchase of the software.

[3] The Adobe Distiller workflow is `.tex` → `dvi` → `.ps` → `.pdf`, conversion to PS through dvips (or dvipsone

2. LaTeX Requirements

It is assumed you have a TeX system up and running on your Windows or Mac platform,[4] and that you are familiar with how to write and compile a LaTeX document and how to create a PDF using Adobe Distiller.

2.1. The atb Package

The atb package, used to build the control panel ease.pdf, requires the Web package and the eForms package, both of which are included with the **AcroTeX eDucation Bundle** (AeB) distribution. Below is a list of other required packages used by the ATB:

1. hyperref: The hyperref bundle should be already on your system, it is standard to most LaTeX distributions.

2. xkeyval: The very excellent package by Hendri Adriaens. This package allows developers to write commands that take a variety of complex optional arguments. You should get the most recent version.

3. xcolor: (optional) An amazing color package by Dr. Uwe Kern. The Web package uses xcolor if present on your system, otherwise, it uses the standard color package. Get a recent version.

4. comment: A general purpose package, Victor Eijkhout, for creating environments that can be included in the document or excluded as comments. A very useful package for LaTeX package developers. This package is distributed with @EASE.

☛ One of the extremely nice features of **MiKTeX** is that it can automatically download and install any unknown packages onto your hard drive, so getting the @EASE up and running is not a problem!

2.2. The atbdb Package

This package is used to create @EASE DB documents. (See 'The @EASE Database' on page 25 for details of usage.) The whole exam assembly system (@EASE) is based on the eqexam package, which is distributed with the AeB as well as with @EASE. Additional LaTeX packages, such as PSTricks, can be included in any particular DB document. An abbreviated manual for eqexam is provided in this manual, see 'The eqexam Package' on page 37.

[4] **Acrobat Professional 7.0** is not available on Linux platforms.

1. The ATB Package

The ATB Package (AcroTEX Test Bundle) is the LaTeX package used to build the control document `ease.pdf` from its source file `ease.tex`. The installation includes the following files:

- `atb.sty`: The style file for `ease.tex`
- `atbdb.sty`: The style file for creating @EASE DB documents, for details of usage of `atbdb.sty`, see 'The @EASE Database' on page 25
- `atb.js`: Special JavaScript commands used by `ease.pdf`
- `eqexam.cfg`: Special configuration file used by the `atbdb.sty`, this "bundles" all the `eqexam` options needed for DB document creation into one option
- `atb_str_us.def`, `atb_str_de.def`, `atb_str_cus_template.def`: For English, German and custom Language localization, respectively

The installation of these files is discussed in the next section.

1. Installation

@EASE uses both the acrotex and eqexam packages, these packages need to be installed, if not already present on your system.[1,2]

☛ The installation of @EASE requires some special steps to perform, details follow. Most modern LaTeX systems require custom packages—ones that are not otherwise available from CTAN—to be by hand in a local TDS tree that you create; MiKTeX refers to these custom TDS trees as *root* directories. For those using MiKTeX help page on this topic,

`http://docs.miktex.org/manual/localadditions.html`

explains how to create a (custom) root directory. Check the documentation for other LaTeX systems (TeX Live, for example) on how to install custom packages. Within your custom root directory, say,

`C:\Local TeX Files\tex\latex`[3] (1.1)

copy `atb.zip` and unzip it. This will create the `atb` folder.

[1] The acrotex bundle is found at `ctan.org/pkg/acrotex`
[2] The acrotex bundle is found at `ctan.org/pkg/eqexam`
[3] The suggested name of the local folder, but you can choose any name.

Unzipping and Installing `atb.zip`. The following are the steps for installation of @EASE:

1. As mentioned above, after you've created your own TDS tree (root directory) and registered it with your LaTeX system, you should have copied `atb.zip` into the `latex` folder, refer to display (1.1) above.

2. Locate the `atb.js` in the `atb` folder and copy it to the user JavaScript folder of Acrobat. Where is this folder? This is a difficult question because its location depends on the version of Acrobat you have. The ultimate answer to this question is in the accompanying document `install_jsfiles.pdf` found in the `doc` folder of this distribution.

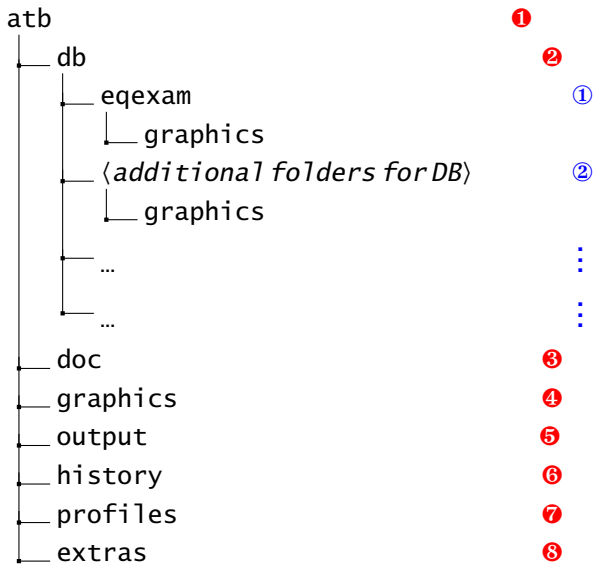
   To verify the correct installation of `atb.js`, close Acrobat and start it again, then look under the View menu (Tools menu for Acrobat 9 or prior) to verify the presence of the sub-menu @EASE Tools. If it is not there, return to step (2) above and check the placement of `atb.js`.

3. Take the file `eqexam.cfg`, found in the `atb` folder and copy/move it to the folder that contains the `eqexam` package.

4. If you are using MiKTeX, or some such system, you need to refresh your database of file names.

2. The `atb` Folder Structure

The following diagram illustrates the folder structure of the `atb` installation, numbered comments follow.



❶ `atb`: The root level of the @EASE distribution. At the top level reside, in addition to its subfolders, the files `ease.tex`, `ease.pdf` and `ease_de.pdf`.

Important: The @EASE Control Document, ease.pdf, should not be removed from the top level of this folder. The reason for this is to maintain the relative relationship between ease.pdf and the database folder db, described next. You can, for example, take the entire atb folder and copy it to another computer hard drive; the system should work as it did on the first computer.

❷ db: This folder holds the database system files for @EASE.

① db/eqexam: This subfolder of db contains the sample DB documents that are shipped with @EASE.

db/eqexam/graphics: The graphics subfolder contains all the graphics files used by the DB document in the eqexam folder.

② db/⟨*additional folders for DB*⟩: All DB documents should be placed in the db folder. Create additional folders as needed using whatever method of organization you desire.

Each DB folder must contain a graphics subfolder where all the graphics files for that folder are required to be kept.

Note: In the most recent version of @EASE, the graphics files may go in any subfolder.

❸ doc: This is the folder that contains atb_man.pdf, this file.

❹ graphics: Folder that contains various graphics files for ease.tex.

❺ output: This is a convenience folder for you to save your assembled exams. After you are happy with your exam, you can then move myExam.tex,[4] and any graphics files the exam uses, to whatever folder you wish.

The assembled exam (default name myExam.tex) can actually be saved anywhere on your hard drive, perhaps in a folder devoted to your current class. The output folder is there simply for your convenience.

❻ history: @EASE has a control for saving the internal IDs of the questions selected for an exam, see the description of the Save button on page 15. These data files (called history files) are saved to the history folder. When using the Load button (described on page 15), browse for this folder and select your history file.

❼ profiles: @EASE has a control for loading and saving class data as described on page 11. The data files (called profiles) are saved to this folder.

❽ extras: If you are using the WinEdt editor (wwww.winedt.com), the extras folder contains some WinEdt macros (ID.edt and isList.edt) for automatically tagging your DB questions. The macros were developed by Jürgen Gilg, with the able

[4] The default name of the exam being assembled. This name can be changed by the user, that's you.

assistance of Aleksander Simonič and Karl Koeller. Refer to the file `ID.edt` for installation instructions. Tagging (or markup) of problems is discussed in Section 4. Use the file ID_test.tex to test the `ID.edt` macro; be sure to read the extensive comments on the limitations of this macro.

3. Options of the `atb` Package

This section can be skipped on first reading. The @EASE distribution comes with the @EASE control panel `ease.pdf` ready to go. Should you want to rebuild the panel—perhaps you wrote your own language localization and want your control panel to reflect your own language—read on.

The file `atb.dtx` is a wrapper of a number of different files used by @EASE, as listed on page 5. In this section, we concentrate on `atb.sty`, the style file used to build the control document `ease.pdf`.

The source file for `ease.pdf`—the main control document of @EASE—is `ease.tex`. The @EASE distribution comes with two versions of `ease.pdf` ready for use:

- `ease.pdf`: The English version of the control document.
- `ease_de.pdf`: The German version of the control document. (If you wish to use the German version, archive `ease.pdf` and rename `ease_de.pdf` to `ease.pdf`.

Because `ease.pdf` already comes compiled, there may never be a need to compile `ease.tex`;[5] in any case, the options for the `atb` package are discussed.

The `atb` options are as follows:

- `lang`: Use the `lang` key to specify the language to be used when `ease.tex` is compiled. Recognized values are `english` (the default), `german` and `custom`. For example, to compile with the `german` language option, we use the `atb` package with the option specification of

  ```
  \usepackage[lang=german]{atb}
  ```

  See Section 4, page 9, for a discussion of how to create a custom language file used by the `lang=custom` option.

- `skin`: Use this option to set the background of the control panel. Currently, there are only two permissible values, `default` and `metal`

  ```
  \usepackage[skin=metal]{atb}
  ```

  If `skin` is not specified, then the default skin is used. New skins may be added in the future.

When compiling `ease.tex`, you also need to specify the driver option of the `Web` package. For users of MiKTEX, for example, you should specify the `dvips` option. For example,

[5] There will be reasons for compiling the `ease.tex` document, these reasons will be discussed later.

```
\usepackage[dvips,rightpanel]{web}
\usepackage[lang=german]{atb}
```

See the preamble of the `ease.tex` to see how these two packages are used. Do not change the `rightpanel` option of the `Web` package, the graphical background assumes the panel is on the right. Also, do not change the following lines either:

```
\margins{.25in}{.25in}{24pt}{.25in}
\screensize{3.5in}{4.75in}
\minPanelWidth{.8in}\panelwidth{.8in}
```

Again, the graphical background assumes these dimensions.

► The `ease.tex` is really not designed to be changed, other than through the language and driver options.

4. Language Localizations

At this writing, @EASE only supports two languages, English and German. When either the `english` or `german` option is the value of the `lang` key of the `atb` package, a language localization file, either `atb_str_us.def` or `atb_str_de.def`, is input when `ease.tex` is compiled.

Take the custom localization file `atb_str_cus_template.def`, make a copy of it, and save it under the name of `atb_str_cus.def`. Open `atb_str_cus.def` in your text editor, and edit the strings in your language. Obviously, save the file under the same name.

Now, edit `ease.tex` to read

```
\usepackage[lang=custom]{atb}
```

and compile the document and build `ease.pdf` using **Distiller**. If all goes well, you will be at ease in your own language customized @EASE.

► Submit your language localization file to me, `dpstory@acrotex.net`, and it will be incorporated into the @EASE distribution, complete with a new language option.

2. The Control Panel for @EASE

The acronym for the AcroTEX Exam Assembly System is @EASE, which is the name of our product. As the name implies, this is an exam assembly system. Fundamentally, there are two components to this system:

1. The document `ease.pdf`: This is the main control of @EASE. You open this document in **Acrobat**, using the controls, you bring up one or more database (DB) documents, select questions from these documents, and finally, let the JavaScript of `ease.pdf` build your exam based on the questions selected.

2. The DB Documents: The DB documents are PDFs that contain typeset versions of exam questions. Only a small DB of questions is shipped with @EASE because we leave it up to the user, that's you, to build your own DB of questions to draw from. Detailed instructions for building your own DB of questions are given later in this document.

The underlying format used for creating your exams and for creating your own DB documents is `eqexam`, another member in the AcroTEX family of education tools.

1. The Methodology

Beginning with **Acrobat 5.0**, JavaScript methods were introduced for creating and manipulating attachments, improvements to these methods continued in **Acrobat Pro 6.0**, and a couple of new methods were added in version 7.0 of **Acrobat Pro**. The JavaScript of @EASE utilizes some of the JS methods from **Acrobat Pro 7.0** to create and manipulate attachments.

You may ask, "What attachments?" Let me explain. The user opens `ease.pdf` in **Acrobat Pro 7.0**. Using the controls, the user opens one or more DB documents into the viewer. See Figure 2.1, page 11, for a visualization. The user browses the DB documents and selects problems by checking the check box corresponding to that problem. When the user clicks on the Get button (described in Section 2) back in the control document `ease.pdf`, the activated JavaScript retrieves the LaTeX source code for the questions selected. The JavaScript then builds an `eqexam` document based on any options taken.

"Where does the JavaScript get the LaTeX source code for the selected problems?" Each DB document, when it is compiled using the style file `atbdb.sty`, has its source file attached to the DB document. The LaTeX source file has markup embedded in it that the JavaScript uses to locate any particular question. The JavaScript then copies the code for the question.

"Where is the `eqexam` document just created stored?" When the user clicks on the Get button, as explained in the previous paragraph, the LaTeX is retrieved and the `eqexam` document is created. The resultant LaTeX document is attached to `ease.pdf`; also attached to `ease.pdf` are any graphics files needed by the selected problems.[1] When the

[1] You can see the attached files by clicking on the "Attachments" tab of the (left) navigation pane of **Acrobat**, while `ease.pdf` is the active document.

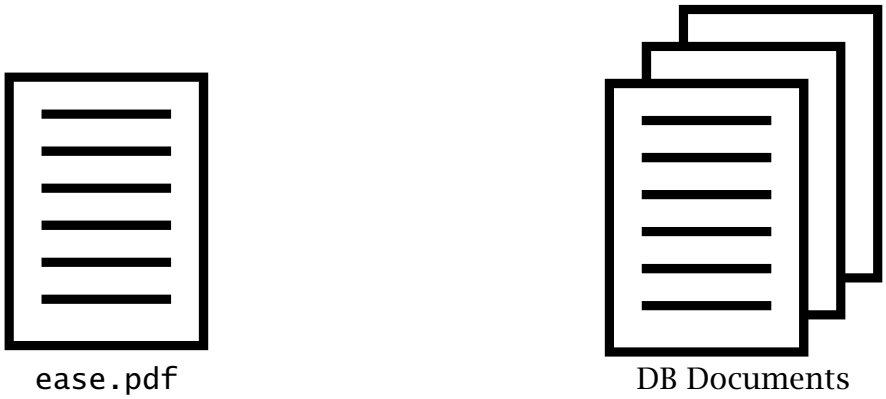


Figure 2.1: @EASE Documents

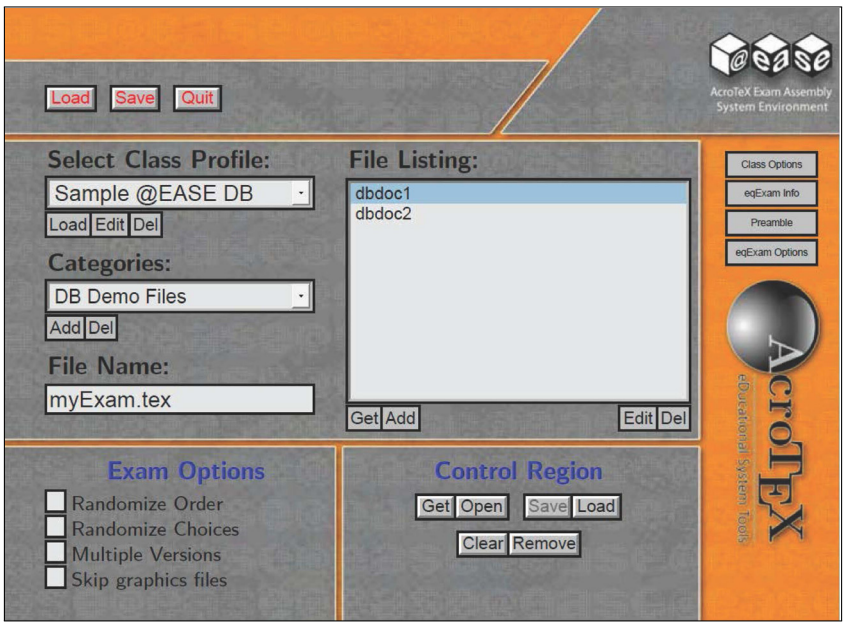


Figure 2.2: @EASE Control Panel

user clicks on the Open button on ease.pdf, your are prompted to save the attachments to a folder.

Cool! :-{)

2. Description of Controls

The central control of @EASE is ease.pdf. This document contains a number of buttons, combo boxes and list boxes. The purpose of this section is to give detailed explanations of these form field elements. Refer to Figure 2.2 on page 11.

The top row of buttons

- Initial Load @EASE Data
  Load @EASE stores and saves its data in the profile folder in the form of FDF

files. When you initially open `ease.pdf`, click on the Load button to load previously saved data.

- Save @EASE Data
  Save After you make any modifications to your file system, click on the Save button to save all changes to your hard drive. Stored data reside in FDF files located in the `profiles` folder.

- Quit @EASE
  Click on the Quit button to close the DB documents that are open; Shift-click on the Quit button to close not only the DB documents, but the `ease.pdf` document itself.

The middle left screen region

- **Select Class Profile:**
  The instructor normally teaches classes covering many different subject areas: Basic Math, College Algebra, Calculus, Linear Algebra, etc.[2] @EASE comes with ten class profiles (or subjects) hard-wired into the system. The items in the combo box represent particular classes or general subjects that you want to categorize. The default names are `Class 1`, ..., `Class 10`. These names can be changed with the Edit. The combo box has three controlling buttons:

  - Load After you select your class profile, click on the Load button to load the data associated with the selected class profile.
  - Edit Click on this button to change the name of the selected class profile name. A response box appears, enter the name you want to give to the class profile in the text field provided; for example, you might type in `Calculus 1`, to represent the first semester of Calculus. Click on `OK` to commit, or `Cancel` (or press the `Esc` key on the keyboard) to cancel the Edit.
  - Del Press the Del button to delete a class profile. Your custom name will be replaced by the default name (`Class #`), all categories associated with that class profile will also be deleted and the file listing will be deleted as well. (The DB files themselves are never deleted.)

- **Categories:**
  The categories are the subtopics associated with the current class profile. For example, if the class profile is `Calculus 1`, you might have as categories the topics of *limits*, *differentiation*, *integration*, and so on. There is no limit to the number of categories (subtopics) you can have.

  The Categories combo box has two controlling buttons:

  - Add Click on the Add button to add a new category to the current class profile. When you click on Add, a response box appears. Enter the new name

[2] Of course, the instructor is not limited to mathematical courses.

into the text field provided. Currently, there is no way to rename a field you have already named. (You would have to delete the category, and click on the Add button to create a new category.)

Changes don't become permanent until the Save button is pressed.

– Del Press the Del button to delete the current category. The file listing associated with the category will also be deleted. After you delete, you may have to click on the class profile Load button to refresh the listing.

Changes don't become permanent until the Save button is pressed.

- **File Name:**
The text field labelled **File Name** is the default name, `myExam.tex`, of the exam to be created. Enter the preferred name of the exam in this field.

The middle right screen region

- **File Listing:**
Each class profile has one or more categories, each category has associated with it a list of one or more files. These files are, of course, DB documents containing questions covering the topics of the current class and category.

The File Listing list box has four controlling buttons.

– Get Click on this button to get the PDFs for the selected DB document.

To get a single file, highlight it by clicking on its name, and press the Get button. To get several files, ctrl-click (or shift-click) on several files to highlight them all, then press the Get button.

– Add Once you have selected your class profile and category, you can then associate DB documents. Click on Add and an Open file dialog appears. You can browse for the file, highlight it, and click on the Open button of the dialog. The base name of the file selected will then appear in your file list box.

Changes don't become permanent until the Save button is pressed.

– Edit When you Add a file, the default name is the base name of the file. You can rename this to something more descriptive. Clicking on the Edit button leads to a response box, type in a new file description, click on OK. The description now appears in the file listing with the base name in parentheses. The Cancel button (or the Esc key) dismisses the dialog without any change. To give a new description, just repeat the above steps. To revert back to the base name only, click on Add and press on OK without entering a new name in the text field.

Changes don't become permanent until the Save button is pressed.

- Del Delete one more files visible in the file list box by highlighting the targeted file(s) by clicking (shift-clicking or ctrl-clicking), then pressing the Del button.

  Changes don't become permanent until the Save button is pressed.

The bottom left screen region

- **Exam Options:**
  There are a several options you can take by clicking on the check boxes provided.
  - Randomize Order: Clicking on this check box tells @EASE to put the selected question in random order.
  - Randomize Choices: Clicking on this check box tells @EASE randomize the choices (alternatives) in any multiple choice questions selected.
  - Multiple Versions: You may be teaching several sections of the same course, or be teaching a course with a large enrollment. In this case, you may want two or more equivalent versions of the same exam, each with a different heading; perhaps, one with Section 001, another with Section 002; or one with Version A and another with Version B. @EASE allows you to create multiple version of an exam, with different headings.

    If you are developing a exam that needs multiple versions, check this option.
  - Skip graphics files: When you click on the Open button any required graphic files are also saved.[3] There may be the occasion that you already have the graphics files in your "save" folder, and do not want to save and re-save them as you, perhaps, revise your selection of questions. Click on this checkbox to skip the saving of the graphics files.

The bottom right screen region

- **Control Region**
  This is the main control region for getting your selected questions, building the LaTeX source file and for saving it to a folder. There are six controlling buttons.
  - Get After you have selected the questions to be included in your exam, click on the Get button to get the source code for the selected questions. This button also builds the LaTeX file, taking into consideration the **Exam Options**. The **Exam Options** should be taken before clicking on the Get, if not, select the **Exam Options** and click on Get again!
  - Open After you have clicked on Get, you are ready to save the LaTeX file to a folder. To do this, click on the Open button. You will be prompted to save each of the graphics files needed for the exam—unless the Skip graphics option is taken—and as well as the exam itself. The LaTeX file will also be loaded into your editor for your review and for compilation.

[3] These would be `.eps` files for the Distiller workflow, or `.pdf/.jpg` files for pdflatex or xelatex.

- Save After you have clicked on the Get button, this button becomes active. When you click on Save, you will be prompted to save the selection of questions to a `history`. Give the file a name—with no extension, one will be supplied—and save it to the `history` folder for later use. This may be useful if you are interrupted in the building of your exam, and wish to save selections made so you can continue at a later time; or if you wish to keep a record of your selections to build an "equivalent" version of your exam at a later time. The data is recovered by the Load button described next.

  The Save button becomes inactive again when you click on the Clear button.

- Load Click on this button to load a file saved by the Save button. When the saved history file is loaded, the DB documents that were in use are loaded, and the selected questions are rechecked programmatically with JavaScript.

  Example. @EASE comes with one FDF that has been saved. Click the Load button and navigate to the `history` folder and select `atbTest.fdf`. After highlighting this file, press the **Select** button on the **Select File Containing Form Data** dialog box. The sample DB files `dbdoc1.pdf` and `dbdoc2.pdf` should open; you can see the selections made by myself at some undeterminate time in the past. At this point you can select more problems, or de-select some of the ones already chosen. Press the Get button, then the Open button to save the new `eqexam` document to your file system.

- Clear This button is a form reset. It will clear all fields back to their default values. This button also makes the Save button inactive.

- Remove @EASE builds the exam and saves it as an attachment to the control document `ease.pdf`; @EASE also attaches any needed graphics files to `ease.pdf`. Clicking this button removes the attachments, presumably, after you have already saved everything.

The right panel region

The right panel has three buttons for entering various information into the exam document being constructed. All the info that goes into the dialog boxes of under these buttons are saved with the Save button.

- Class Options Once you have loaded your profile, click on the Class Options button to set paper size and point size. The `default` settings will not generate any optional arguments to the `article` class; hence, the paper and point size are whatever is the default for your TEX system. Pressing OK automatically saves these settings along with your profile settings. See 'Class Options' on page 16 for more details.

- eqexam Info An `eqexam` document has certain key-value pairs that are used to build an exam document, the keys are `\title`, `\subject`, `\author`, `\university`, `\date`, `\duedate` and `\keywords`. Clicking on the eqexam Info button gives you

access to entering the values for these keys. This information is saved by the Save button as part of the class profile.

See Section 2.2, 'eqexam Info' on page 17, for a detailed description of these fields.

- Preamble Click this button to open a dialog box in which you can enter any additional LaTeX commands that should go in the preamble of the exam document you are building. Of course, when the exam document is built, you can also edit it directly in your favorite editor as well. See 'Preamble' on page 20 for more details.

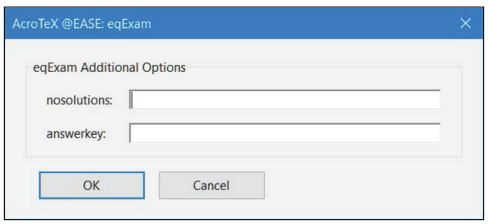


Figure 2.3: eqExam Additional Options

- eqExam Options Click this button to open a dialog box in which you can enter additional `eqexam` options for the exam document you are creating. The default options are

```
%\usepackage[forpaper,pointsonleft,nototals,answerkey]{eqexam}
\usepackage[forpaper,pointsonleft,nototals,nosolutions]{eqexam}
```

This dialog box allows you to add onto these default options. One application is to pass a configuration file option (`myconfigi-myconfigvi`). You can have one configuration files for the `nosolutions` option and another for the `answerkey` option. Another possibility is to use the `cfg` option for importing a CFG file; for example, if you enter 'nosolutions: `cfg=nosolutions`' and 'answerkey: `cfg=answerkey`', then when the `myExam.tex` file is built and saved, the options now read:

```
%\usepackage[forpaper,pointsonleft,nototals,answerkey,
%    cfg=answerkey]{eqexam}
\usepackage[forpaper,pointsonleft,nototals,nosolutions,
    cfg=nosolutions]{eqexam}
```

When you compile `myExam.tex`, `eqexam` then looks for either `nosolutions.cfg` or `answerkey.cfg`, depending on the line used. In the listing, the lines are wrapped around to the next line for display purposes.

2.1. Class Options

Use the Class Options dialog box to enter of the paper and point size options for the exam being assembled. See Figure 2.4.

Through this dialog, you can set the paper and pointsize:

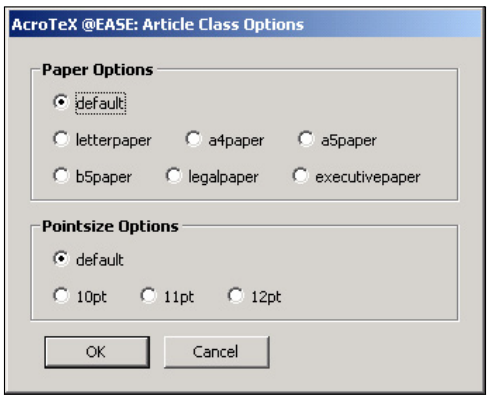


Figure 2.4: The Class Options Dialog

- **Paper Options**: The options are `default` (the paper size that is the default for your TEX system), `letterpaper`, `a4paper`, `a5paper`, `b5paper`, `legalpaper` and `executivepaper`.
- **Pointsize Options**: The options are `default` (the point size that is the default for your TEX system), `10pt`, `11pt` and `12pt`.

These selections are saved with the profile of the course.

2.2. eqexam Info

The eqexam Info fields dialog, obtained by click on the eqexam Info button, consists of ten text fields. The information you put in these fields are inserted into your LATEX document; therefore, you should input LATEX markup. In the descriptions below, refer to Figure 2.5 on page 18. We describe the order they are listed in the dialog.

1. Course: Place the course name here, with possibly a section number; for example `Calculus 1`. This entry appears in the title of the document and in the running header. There is an optional leading argument you can also specify. The optional argument is delimited by brackets. If the optional argument is specified, this argument will be used in the running headers on subsequent pages. For example, `[C1]Calculus 1`, titles the course to be 'Calculus 1' and eqexam will use 'C1' as part of the running header.

   The values of the Course field and short version are held in the two commands `\websubject` and `\shortwebsubject`. (The names of these commands reflect the close relationship eqexam has with the Web package. There is a similar naming convention so the two packages are harmonious, one with the other.)

2. Test #: The entry in this field defines the test number of the test. For example, the entry might be a `1` or `3-Version B` or `Final Exam`. The value entered in this field is saved in the text macro `\nExam`. The command `\nExam` can be used in other fields to refer to the test number.

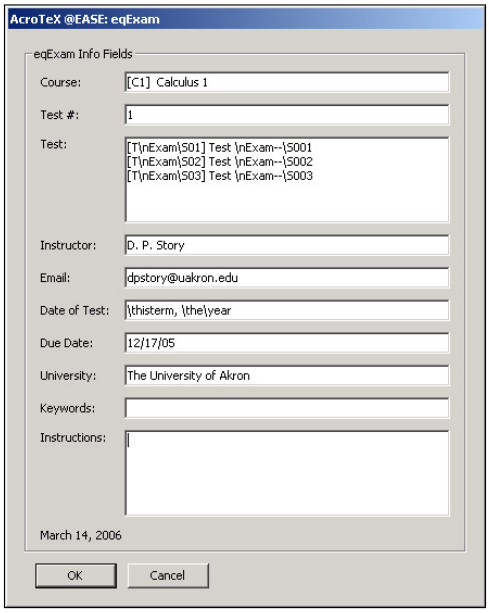


Figure 2.5: eqexam Info Dialog

3. Test: The entry in this field determines the exam title, see Figure 2.5.

   When building a single exam, enter the name of the exam in this field, for example, `Test \nExam`, or `Exam \nExam` or even `Homework \nExam`. The title also appears in the running header of the document and there is, therefore, an optional argument for placing something shorter in the running header. For example, we could specify `[T\nExam]Test \nExam`, and `T1` (assuming `\nExam` expands to `1`) appears in the running header.

   When building an exam with multiple versions, place one title per line for each of the versions. For example,

   ```
   [T\nExam\S01] Test \nExam--\S001
   [T\nExam\S02] Test \nExam--\S002
   [T\nExam\S03] Test \nExam--\S003
   ```

   This is what is actually shown in Figure 2.5.

   If there are multiple listings in the title field, but the Multiple Versions checkbox is not checked, @EASE takes the first title as the title of the exam.

   The values of this field and the value of its optional argument are held in the macros `\webtitle` and `\shortwebtitle`.

4. Instructor: The name of the instructor or instructors. A large number of instructors may require a redefining of the `\maketitledesign` command to fit them in.

5. Email: By default, this field only appears when you use the `coverpage` option. The `\maketitledesign` command can be redefined to incorporate it in otherwise.

   The value of this field is held in the macro `\webauthor`.

6. Date of Test: In this field, place the date of test. This entry appears, by default, as part of the `\maketitle`. In Figure 2.5, this entry is `\thisterm, \the\year`. The command `\thisterm` is an `eqexam` command that expands to 'Fall', 'Spring' and 'Summer', depending on the date. This is the form I prefer in my tests. The command `\thisterm` can be redefined to yield the names of the term (semester, quarter) name at your school. Or, you can simply put in a date.

   The value of this field is held in the command `\@date`, and defaults to the date of compilation when no date field is specified.

7. Due Date: The value of this field is typically a date. For tests, I put in the acutal test date (because I use `\thisterm, \the\year` for the date field). The value of this field becomes to value of the text macro `\theduedate`, which can, in turn, be used in other fields.

   I use `eqexam` to typeset homework assignments, and I use this field to indicate the due date for the assignment. I have a special configuration file for homework assignments in which the `\maketitledesign` macro is redefined to include the due date, `\theduedate`.

8. University: The title of your university, college, high school, etc. In the default layout of an `eqexam` exam, the value of the university key only appears when you use the `coverpage` option. Again, in other cases, if you want to use this entry, you need to redefine `\maketitledesign`.

   The value of this field is held in the command `\webuniversity`.

9. Keywords: The keywords field is only relevant if you convert your newly created document to PDF using one of the options `pdf`, `links`, `online` or `email`.

   When I use `eqexam` to publish homework assignments posted to the Web, I use this field to document the assignment: `Homework \nExam, due on \theduedate`, for example.

10. Instructions: A test or homework assignment usually has initial instructions, such as "Solve all these problems without error, or fail trying." You can enter your instructions through this field. Again, keep in mind, you are entering LaTeX markup. So an instruction might be

    ```
    Solve these problems, passing is the $80^{\text{th}}$ percentile.
    ```

    This field too has an optional argument, delimited by brackets. By default, the value of this field appears as follows:

Instructions. Solve these problems, passing is the 80th percentile.

The word “Instructions” above can be replaced by whatever word you wish, thus, if you specify an optional argument, like so, then

```
[Global Instructions.]Solve these problems, passing is
the $80^{\text{th}}$ percentile.
```

the instructions for the exam appear like so,

Global Instructions. Solve these problems, passing is the 80th percentile.

2.3. Preamble

Use the Preamble dialog box to enter any additional packages and/or commands you want to include in the preamble. LaTeX commands are entered in the large multiline text window.

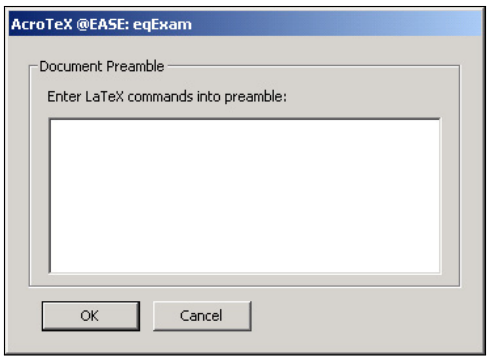


Figure 2.6: The Preamble Dialog

The entries in this dialog are saved with the profile of the course.

3. The @EASE Toolbar Buttons

When you copied the `atb.js` file in the user JavaScript folder, you installed the @EASE Toolbar Buttons. Use of the toolbar buttons depends on whether the version of Acrobat you use is less than 10 or 10 or greater.

3.1. Toolbar Buttons: Acrobat earlier than Version 10

If you are using Acrobat version 9.x or earlier, to use the toolbar buttons follow these steps:

1. Start the **Acrobat** application, if it is already up and running, close all PDF documents open in the viewer.

2. Click on the Tools menu (View menu for Acrobat 10) item on your **Acrobat** menu bar, from the drop down menu list, select @EASE Tools.

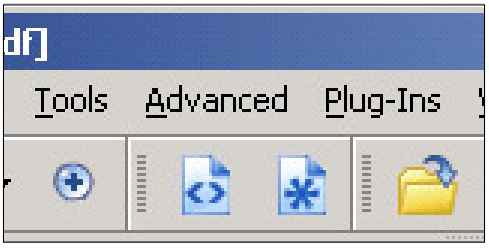


Figure 2.7: @EASE Toolbar Buttons

3. A group of two buttons should appear on your toolbar.

Figure 2.7 depicts the toolbar buttons.[4] The button on the left toggles through the @EASE documents open in the viewer. The button on the right brings the @EASE control panel to the top of the stack of documents. These buttons are useful when trying to navigate when the control panel and several DB documents are open in the viewer.

To load the @EASE control panel (`ease.pdf`), press the button with the * on it (the one on the right, as depicted above). The first time you press this button, however, you get an alert message saying "Cannot find the @EASE Control Panel, `ease.pdf`, try manually loading the file." Navigate to the `atb` folder, locate `ease.pdf`. When `ease.pdf` it will create a global variable that is used to re-load the file using the toolbar buttons.

3.2. Toolbar Buttons: Acrobat 10 or later

The user interface for Acrobat X (Acrobat 10) was re-worked by Adobe engineers. The Tools menu no longer exists, so the @EASE Tools menu was moved to the View menu. To use the toolbar buttons follow these steps:

1. Start the **Acrobat** application, if it is already up and running, close all PDF documents open in the viewer.

2. Click on the View menu item on your **Acrobat** menu bar, from the drop down menu list, select @EASE Tools. The @EASE control panel (`ease.pdf`) opens, or an alert box opens asking you to load the file manually. Once `ease.pdf` is loaded for the first time, its path is remembered and it will open automatically when you select the @EASE Tools menu item.

3. Click on the Tools button on the toolbar, and the Tools Panel opens. If there is a document open in Acrobat (`ease.pdf`, for example) you should see the `Plug-In Add-on Tools` item in the panel. Open it to see the two @EASE toolbar icons. See Figure 2.8 below.

4. If you are using Version 10.1 (or later), drag the two icons to the toolbar. (See Figure 2.9) The Tools Panel can then be closed, and the system works as described in Section 3.1, page 20. If you are using only Version 10 (update to Version 10.1,

[4] The buttons shown have been placed near the Tools menu (View menu for Acrobat 10). They may not be in this position when they are first opened. Move them to any convenient location you wish.

please), the Tools panel needs to be left open so that the @EASE Toolbar Buttons are visible, see Figure 2.8.

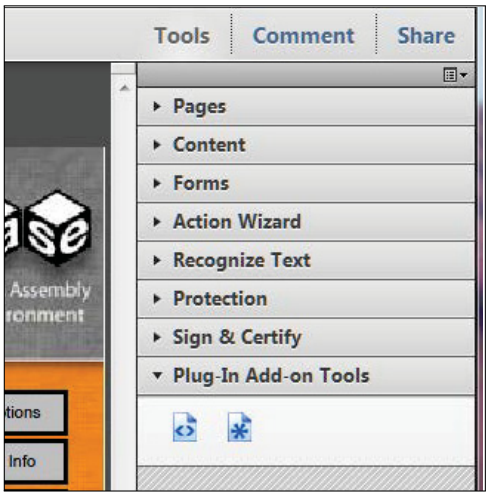


Figure 2.8: @EASE Tools Panel

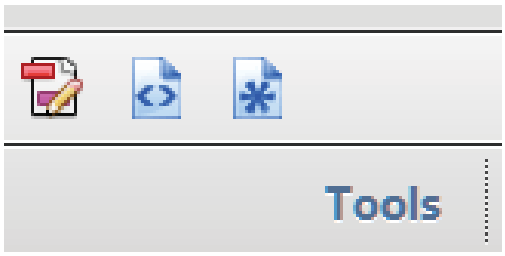


Figure 2.9: @EASE Toolbar Buttons, Version 10.1

Figure 2.8 depicts the toolbar buttons. The button on the left toggles through the @EASE documents open in the viewer. The button on the right brings the @EASE control panel to the top of the stack of documents. These buttons are useful when trying to navigate when the control panel and several DB documents are open in the viewer.

The @EASE Tools can be made to appear in the toolbar of Acrobat X and Acrobat DC, each requires its own steps to accomplish.

4. Building your First Exam

In this section we walk-through the process of assembling an exam, we also discuss post-assembly processing.

4.1. A Walk-through

1. Begin by open the @EASE icon toolbar set by selecting `Tools > @EASE Tools`, or `View > @EASE Tools` (for Acrobat 10 or later),

2. Open `ease.pdf` in **Acrobat Pro 7.0** or later by clicking on the @EASE icon with an * on it, this should load `ease.pdf`.[5] For **Acrobat Pro 10.0** or later, `ease.pdf` opens in Step 1, automatically; however, it may have to be manually opened the first time.

3. Click on the Load button in the upper left-corner.

[5]The path to the `ease.pdf` needs to be established. If the * does not open `ease.pdf`, by double clicking on `ease.pdf` in Windows Explorer. The path is then remembered the next time you click on the * button, the button should open `ease.pdf`.

4. Next, from the Select Class Profile drop-down combo box, select a class profile. (When you first get @EASE, you will have one class profile predefined.)

5. Now from the Categories drop-down combo box select a category. You'll notice that a file listing appears in the "File Listing" list box.

6. Highlight one or more of these files, and click on the Get button under the File Listing list box. The files will be loaded into the **Acrobat** viewer.

7. You can return to `ease.pdf` and choose a different category from the Categories combo box, then choose more files if you wish. These combo boxes are convenient for organizing the files of questions in your database of `eqexam` questions.

8. Now that you have loaded one or more DB documents into the viewer. Peruse them, find some questions you want on your first exam. Select the questions by clicking on the checkbox to the left of the problem number. (You can move between DB documents by clicking on the Toggle @EASE button that you opened on your toolbar. See Section 3, "The @EASE Toolbar Buttons", on page 20.)

   Some problems have two boxes, one above the other. This means that this question has multiple versions of the same type of question. Put your cursor over the lower box, a frame should appear to indicate the number of equivalent questions. If it says there are 4 questions of the same type, you can enter into the lower box an integer n, $1 \leq n \leq 4$, for example. If you enter a 2, @EASE will choose two of the questions. The selection of the number of equivalent questions will be random. If no integer is entered into the box, the first of the equivalent questions (usually the one that is typeset in the DB document) is chosen.

   The selected questions can come from any of the DB documents open in the viewer.

9. When finished with selecting questions, return the control document `ease.pdf`.

10. Fill-in the Info Fields and check the Class Options. (This step can be done much earlier in the process of building your exam.)

    (a) Fill-in the Info Fields: Click on the eqexam Info button, and fill in the information in the large dialog. Section 2.2 explains these fields in detail.

    The contents of these fields are saved with class profile. In the future when you initially load your profile data, the info fields are restored.

    After you fill in the info, click on the Save button in the upper left corner to save these entries.

    (b) Click on the Class Options button in the right navigation panel. Choose your paper and point size options. These are saved automatically when you click on the OK button.

11. Select any of the options Randomize Order, Randomize Choices, Multiple Versions or Skip graphics files.

12. With your DB documents still open in the viewer, and your selections still checked. Click on the Get button in the Control Region. This starts up JavaScript that searches through the DB files opened in the viewer, extracts selected questions from the source, builds the LaTeX source file of the eqexam.

13. As the last step, click on the Open button in the Control Region. You will be prompted to save the exam along with any graphics files that the JavaScript determined is needed. The exam file `myExam.tex` (the default name) will open up in your default TeX editor. If luck is with you, you can LaTeX the document, and *le voilà*, *votre examen*, *Madame et/ou Monsieur*!

☛ Once you assemble your document, you might save your choices using the Save button in the **Control Region**. You might want to go back to the DB documents to add in a few more questions, or remove choices made earlier.

4.2. Post Assembly Processing

Once the exam is built, there may be a need to tweak the source:

- assign points for each problem
- adjust the vertical space allowed for the problems
- rearrange the questions slightly to fit better on the pages
- possibly delete questions for various reasons: too many questions, question too hard, too easy
- When there are multiple versions to a problem, the JavaScript chooses the version to use by randomly assigning a value for the first parameter of `\selectVersion`. For example, in a DB document you might have `\selectVersion{}{3}`, the JavaScript might choose variation 2, so in the assembled document you'll see the command `\selectVersion{2}{3}`. You can change this selection from within the source file of the assembled exam, for example, you might change this to `\selectVersion{3}{3}`.
- change some of the default eqexam options.

3. The @EASE Database

@EASE comes with a small collection of DB documents that can be used to become familiar with how the system works, and with how DB documents are marked up. No large DB of questions is provided.

Members of the ASDT debated over whether to attempt to develop a large database of problems, over a wide variety of topics, and over several languages. The cost of @EASE would be measured in the hundreds or thousands of dollars.

My own feeling is that questions are a very person thing, each must be chosen to be consistent with ones own teaching style, with content of the course, and with the notation used in the text or in lecture. We decided, therefore, to provide tools to assemble an exam from the user's personal DB documents.

1. The DB Document

Important. When you compile your DB document and convert it to PDF, you must bring that newly created PDF into Acrobat, some JavaScript will execute and attach the source file. Verify that the source file is attached, then save the new DB document.

Figure 3.1 shows the recommended preamble for an @EASE DB document. The standard LaTeX packages are article class, amsmath and graphicx.[1] Next comes the eqexam package with the special option `atbdbopts`; *this option is required*. Finally, there is the atbdb, a special package for @EASE.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage[atbdbopts]{eqexam}
\usepackage{atbdb}

\subject[C1]{Calculus I}
\title[Diff]{Differentiation}
\author{D. P. Story}
\keywords{}
\university
{%
      NORTHWEST FLORIDA STATE COLLEGE\\
        Department of Mathematics
}
\email{dpstory@uakron.edu}

\begin{document}
```

Figure 3.1: Preamble of DB Doc

► **pdflatex/xelatex Users.** The same code above works for these two workflows. The eqexam package automatically detects the use of pdflatex and xelatex.

[1] The graphicx package is only needed when there are graphics inserts in the document.

The document information key-value pairs (lines 7–17 in Figure 3.1) follow the inputting of the class and the package files. The two keys \subject and \title are used in the title heading of the DB document and they identify the **Topic** and **Subtopic** of the DB document. The \author key identifies the author or authors of the document; the value of this key appears in the second line on the right.

The rest of the keys are optional and can be used for documentation purposes. The value of \keywords will appear Keywords fields of the document properties of the PDF.

2. Passing preamble material to myExam.tex

A particular DB document may have some localized definitions the problems use. When you select problems that depend on these definitions, the same problems in myExam.tex do not compile. The solution is to either find the missing definitions and copy them into the myExam.tex or to use the \bPreamble/\ePreamble command pair.

```
...
\bPreamble
    ⟨tex_code⟩
\ePreamble
...
\begin{document}
...
```

The ⟨*tex_code*⟩ is any LaTeX markup that you want transferred to your new exam document, generically named myExam.tex. Any material outside the command pair is *not transferred*. Only one \bPreamble/\ePreamble command pair is recognized and should be placed in the preamble.

3. Instructions and Commenting

Following \begin{document} comes the instructions environment. For a DB document, use the instructions environment to generally describe the contents of the document. For example, Figure 3.2 describes the contents of a DB document corresponding to Calculus I.

```
\begin{document}

\begin{instructions}[Description:]
This document contains beginning topics in Calculus I:
functions, limits, continuity, definition of derivative
and elementary applications.
\end{instructions}
```

Figure 3.2: Instructions

Throughout the DB document, you can use the eqComments

☛ Any commenting done within a DB document should appear *outside* the scope of the markups \ID/\endID, described in Section 4.

```
\begin{eqComments}[Description:]
The next three questions on integration are more
difficult.
\end{eqComments}
```

Figure 3.3: Commenting in a DB Document

4. Tagging the problems

The key to the success of this exam assembly system is the markup of the DB documents, also called *tagging the problems*. Correct tagging leads to correct assembly; incorrect tagging, leads to confusion and error.

☛ See the three documents `dbdoc1.tex`, `dbdoc2.tex`, and `dbdoc_nodistiller.tex` for examples of the markup.[2] These three files also have comments with additional details.

There are four markup (or tagging) commands that are used:

1. `\ID`/`\endID`: This pair encloses an entire problem. Syntax:

   ```
   \ID*{⟨unique_tag_name⟩}
   ...
   \endID
   ```

   The first parameter '*' is optional and is used to signal that the question is a problem with *selectable parts*. The second parameter, is a unique identification tag name, consisting of numbers and letters.

   The `\ID` command generates a check box in the left margin. You check this box to select the question. The `\ID*` command generates the check box, and a small text box just below it. Entering a positive integer, n, in this box tells the JavaScript of @EASE to randomly select n of the parts. If left un-filled, it is expected that you would individually select the parts you want to appear on your exam.

2. `\PRT`/`\endPRT`: For a problem with *selectable parts*, this pair encloses an entire part (`\item` itself and its solution). The syntax is

   ```
   \PRT{⟨unique_tag_name⟩}
   ...
   \endPRT
   ```

   The command `\PRT` has one argument, a unique tag name consisting of numbers and letters.

   `\PRT` creates a check box around the item label. Check the box to select the part for your exam.

[2] `dbdoc_nodistiller.tex` is a sample DB file set up for use by `pdflatex` or `xelatex`. Graphical files are PDF files, not EPS files as they are for Distiller workflow.

3. `\selectVersion` This command is used for documenting the number of variations on a problem or part of a problem.

   `\selectVersion{}{⟨number_variations⟩}`

   The second parameter is the number of variations of this problem. The first parameter is typically left empty. The JavaScript of @EASE actually inserts the first parameter for you.

4. `\uses`: Some problems require LaTeX packages beyond the ones that are input by default. The example that comes to mind is a figure or diagram drawn by PSTricks. If a problem has special needs, use the `\uses` command to specify a comma-delimited list of the packages needed. For example,

   ```
   \ID{atb43434}\uses{pstricks,pst-node,pst-plot}
   ```

   This markup should typically follow the `\ID` or the `\PRT` markers. If a package requires special options of a package, enclose the options in brackets and in front of the package name, for example,

   ```
   \ID{atb43434}\uses{[matrix,ps,color,line,graph]xy,...}
   ```

   When the text document, `myExam.tex`, is assembled, the JavaScript of @EASE eliminates duplicate references to the same package, takes the union of all package options for each package, and eliminates duplicate package options as well.

☛ The `\uses` command must immediately follow the `\ID` or the `\PRT` command. Following `\uses`, the rest of the line should be blank. The JavaScript removes the `\uses` commands and its arguments when the exam document is assembled.

Each of these four markups is discussed and illustrated in the subsequent sections.

4.1. Question Types and How to Tag Them

In this section we take a survey of the types of questions and how to mark them up.

- **Questions with No Parts, No Variations**

This type of question is the easiest to handle. Enclose the question with the `\ID`/`\endID` pair. A single example will suffice, see Figure 3.4 on page 29.

The optional argument of the `problem` environment is the number of points assigned to this problem. This number would have to be adjusted once you assemble your exam so that the sum of all points equals the total points of the exam. The optional argument of the `solutions` environment is the amount of vertical space to leave for the solution to this problem. This too may have to be adjusted after assembly.

Note the presence of the `\ID`/`\endID` pair, and the id tag name, `atb2342`, in this example.

```
\ID{atb2342}
\begin{problem}[5]
This is a problem that you need so solve.
\begin{solution}[1in]
This is a solution to the above problem.
\end{solution}
\end{problem}
\endID
```

Figure 3.4: Problem, No Parts, No Variations

- **Questions with No Parts, with Variations**

The markup is the same as in the previous section, additionally, the `\selectVersion` command is used. See Figure 3.5 on page 29.

```
\ID{atb2343}
\selectVersion{}{3}
\begin{problem}[5]
Who was the \vA{first}\vB{second}\vC{third} President
of the United States?
\begin{solution}[1in] The \vA{first}\vB{second}\vC{third}
President of the United States was \vA{George Washington}%
\vB{John Adams}\vC{Thomas Jefferson}.
\end{solution}
\end{problem}
\endID
```

Figure 3.5: Problem, No Parts, with Variations

Note the enclosing `\ID`/`\endID` matched pair, with a unique tag name. Following the `\ID` comes the `\selectVersion` command. It is important to include `\selectVersion` between the markers, and before the beginning of the problem. The commands `\vA`, `\vB`, `\vC`, etc., are used to enter the variations in wording and answer. `eqexam` also generates environments `\begin{verA}`/`\end{verA}`, etc., for enclosing larger amounts of material. In this example, we did not use the environments.

- **Questions with Parts, No Variations**

We use the `problem*` environment to enclose a problem with multiple parts. Figure 3.6 on page 30 illustrates the syntax.

- **Questions with Parts, with Variations**

We use the `problem*` environment to enclose a problem with multiple parts. Figure 3.7 on page 30 illustrates the syntax. The `\selectVersion` states that each part has two variations. You can mix in a part that has no variations.

```
\ID{atb1788}
\begin{problem*}[5ea]
Solve for $x$ in each of the following:
\begin{parts}
\item Solve for $ 2x + 1 = 5 $.
\begin{solution}[1in] Obviously, $ x = 2 $.
\end{solution}
...
\item Solve for $ 2x^3 + 1 = 17 $.
\begin{solution}[1in] Obviously, $ x = 2 $.
\end{solution}
\end{parts}
\end{problem*}
\endID
```

Figure 3.6: Problem, with Parts, no Variations

```
\ID{atb8817}
\selectVersion{}{2}
\begin{problem*}[5ea]
Solve for $x$ in each of the following:
\begin{parts}
\item Solve for $ \vA{2x + 1 = 5}\vB{3x + 1 = 10} $.
\begin{solution}[1in] Obviously, $ \vA{x = 2}\vB{x=3} $.
\end{solution}
...
\item Solve for $ \vA{2x^3 + 1 = 17}\vB{2x^3 + 1 = 3} $.
\begin{solution}[1in] Obviously, $ \vA{x = 2}\vB{x=1} $.
\end{solution}
\end{parts}
\end{problem*}
\endID
```

Figure 3.7: Problem with Parts, with Variations

If not all parts have exactly the same number of variations (excluding the case of no variations), then you need to put a `\selectVersion` in front if each part. This may be a good practice anyway. See Figure 3.8.

Notice, in Figure 3.8, the `\selectVersion` has been removed just before the opening problem statement, there is no variation of wording, so it is really not needed here. The other parts, however, have their own `\selectVersion`, the first one has three parts, the other two have only two. Placing `\selectVersion` in front of each enables the JavaScript to calculate the version of the part, which is selected at random. When you use `\forVersion` at the top of your file, this command enables `eqexam` to calculate the correct version to take.

```
\ID{atb8817}
\begin{problem*}[5ea]
Solve for $x$ in each of the following:
\begin{parts}

\selectVersion{}{3}
\item Solve for $ \vA{2x + 1 = 5}
\vB{3x + 1 = 10}\vC{4x + 1 = 13} $.
\begin{solution}[1in]
    Obviously, $ \vA{x = 2}\vB{x=3}\vC{x=3} $.
\end{solution}

\selectVersion{}{2}
\item Solve for $ \vA{4x^2 - 1 = 3}
\vB{5x^2 - 1 = 19} $.
\begin{solution}[1in]
    Obviously, $\vA{x=\pm1}\vB{x=\pm2}$
\end{solution}
\end{parts}
\end{problem*}
\endID
```

Figure 3.8: Problem with Parts and unequal Variations

- **Questions with Selectable Parts, with Variations**

The individual parts of a multiple part question can be made selectable by enclosing each part with a \PRT/\endPRT pair. The apbdb packages puts a checkbox around the part number of each selectable part. A red border indicates that the part has multiple variations while a blue color means that the part has no variations.

For questions of this type, there are two boxes in the left margin: (1) the one on top is the usual checkbox to indicate that you are choosing this question; (2) the lower box is a text field, when you enter an positive integer, n, in this field, the JavaScript will randomly choose n questions from the selectable parts. Entering a number greater than the number of parts available is equivalent to selecting all of them. When this text field is left empty, then it is expected that you individually select the parts you want.

The following example, Figure 3.9, is the same as Figure 3.8 on page 31, but the parts are made selectable.

Here, we enclose each part with a \PRT/\endPRT pair, with an unique tag name. Now, after the file has been compiled, each part is individually selectable.

- **Selecting Reference Charts, Tables, and Such**

Sometimes, in an exam, the instructor provides resource material in the form of a chart, a table of information, a graph, and so on. Such reference elements normally do not logically fit into a problem environment, but are provided for the student to refer to. Placement of such material is up to the instructor, perhaps at the beginning or at the end of the exam.

```
\ID{atb8817}
\begin{problem*}[5ea]
Solve for $x$ in each of the following:
\begin{parts}

\PRT{3234}
\selectVersion{}{3}
\item Solve for $ \vA{2x + 1 = 5}\vB{3x + 1 = 10}\vC{4x + 1 = 13} $.
\begin{solution}[1in]Obviously,
$\vA{x = 2}\vB{x=3}\vC{x=3}$.
\end{solution}
\endPRT

\PRT{6345}
\selectVersion{}{2}
\item Solve for $ \vA{4x^2 - 1 = 3}\vB{5x^2 - 1 = 19} $.
\begin{solution}[1in]Obviously,
$\vA{x=\pm1}\vB{x=\pm2}$
\end{solution}
\endPRT

\PRT{1345}
\selectVersion{}{2}
\item Solve for $ \vA{2x^3 + 1 = 17}\vB{2x^3 + 1 = 3} $.
\begin{solution}[1in]Obviously,
$ \vA{x = 2}\vB{x=1} $.
\end{solution}
\endPRT

\end{parts}
\end{problem*}
\endID
```

Figure 3.9: Problem with Selectable Parts and unequal Variations

In a DB document, such reference elements can be placed in an `eqComments` environment; see Figure 3.10 for a sample listing. The instructor can develop a number of different selectable reference elements and have them in a DB document. These would then be available for insertion into an @EASE exam at assembly time.

4.2. Multiple Choice with \eFreeze

With regard to the construction of a multiple choice question, a common strategy is to include amongst the various choices a "None of these" choice.[3] Normally, such special choices are listed last in the list of choices. A problem occurs when the instructor assembles the exam using @EASE and specifies the Randomize Choices option on the control panel, all choices are randomized.

To force the placement of special alternatives, such as "None of these" to be at the

[3] "None of these" is sometimes abbreviated by "n.o.t."

```
\ID{atb020506104934}

% Horizontal rule to separate the table below, from the other material
\makebox[\linewidth][c]{\rule{.76\textwidth}{.4pt}}

\begin{eqComments}[Recall:]
Use the following facts freely throughout the exam.
\begin{equation*}
    \sin^2x + \cos^2x = 1, \forall x
\end{equation*}
\end{eqComments}
\endID
```

Figure 3.10: Inserting a Reference Table

end of the list of choices even under the conditions of randomization, a special markup is recognized, the `\eFreeze` command.

```
\ID{apb23423}
\begin{problem}[5]
In what ancient year did Columbus sail the ocean blue?
    \begin{answers}{4} % specify tabular with 4 columns
    \bChoices
        \Ans0 1490\eAns
        \Ans0 1491\eAns
        \Ans1 1492\eAns
    \eFreeze
        \Ans0 None of these\eAns
    \eChoices
    \end{answers}
\begin{solution}
    Yes, Columbus sailed the ocean blue in 1492.
\end{solution}
\end{problem}
\endID
```

Figure 3.11: The use of `\eFreeze`

Figure 3.11 illustrates the use of `\eFreeze`. In a multiple choice question, place `\eFreeze` before any choices you want to be protected against randomization. Any choices that follow `\eFreeze` will appear in the same order they appear in the DB document.

After assembly, the problem in Figure 3.11 might appear in your `eqexam` document as in Figure 3.12. Notice that the first three choices have been randomized, but the "None of these" alternative is still the last alternative listed.

```
\begin{problem}[4]
In what ancient year did Columbus sail the ocean blue?
    \begin{answers}{4}
    \bChoices
        \Ans0 1491\eAns
        \Ans1 1492\eAns
        \Ans0 1490\eAns
    \eFreeze
        \Ans0 None of these\eAns
    \eChoices
    \end{answers}
\begin{solution}
    Yes, Columbus sailed the ocean blue in 1492.
\end{solution}
\end{problem}
```

Figure 3.12: `\eFreeze` Illustrated

5. Additional Guidelines for a good DB Document

In order to build a DB document that can successfully be scanned by the JavaScript of `ease.pdf`, the following rules are recommended.

- In the earlier version of any graphics files must be kept in the `graphics` subfolder of the DB document's folder; in this version, they may be kept at the top level or in any subfolder of the DB document's folder,

  The argument of the `\includegraphics` command must look like this:

```
1 \includegraphics[height=.5in]{\atbpath/dpsweb.eps}
2 \includegraphics[height=.5in]{./graphics/dpsweb.eps}
3 \includegraphics[scale=.5]{./myGraphics/myFig.eps}
4 \includegraphics[width=2in]{./myGraphics/myFig.pdf}
5 \includegraphics{myFig.pdf}
```

  The optional argument above is, of course, optional. Note the presence of the command `\atbpath` in line 1, this command was *required*, but is now *optional*. The `\atbpath` command is defined in the `atbdb` package and expands to `./graphics`, referring to the `graphics` subfolder. The command is still supported, but is not required. Paths to the graphics files can be entered as shown in lines 2–5. For the Distiller PDF creator, use only EPS files; for `pdflatex` or `xelatex` use any graphic format supported by those applications; include into the file using `\includegraphics`. (Other methods of inclusion are not supported.)

  The @EASE JavaScript goes to the referenced folder and copies it to the @EASE Control Panel. The graphics files are then saved into the same folder as `myExam.tex`. When `myExam.tex` is LaTeXed. JavaScript also removes the path to the graphic files so they will be found when `myExam.tex` is LaTeXed; for example, within the source

file,

```
\includegraphics{./mygraphics/mypdfs/myGraph.pdf}
```

is changed to

```
\includegraphics{myGraph.pdf}
```

if all works as it should.

To repeat, graphics files for a DB document must be in the folder containing the DB document, or a subfolder.

- Avoid the use of custom macros in the preamble, these macros will not be recognized when you compile the assembled document, `myExam.tex`.

  If you must have your own private macros to minimize the amount of typing, put all your macros in a `.sty` file for that folder, and include it into your DB document with a `\usepackage` command.

  Using the `Preamble` button, place the same `\usepackage` command in the dialog text box provided. Assuming your little macro package is in the LaTeX search path, and that it has a unique name, the package will be found when `eqexam.tex` is compiled. For portability of the source file, you might have to copy this package to the folder where you saved `myExam.tex`.

- The DB document source file should be clean, a commented out `\ID` will be detected as part of the database and could cause problems, especially if its tag name is the same as an `\ID` that is not commented out.

- JS does not search for `\end{document}` so anything after that marker may be scanned by JS also.

- The pairs `\ID`/`\endID` and `\PRT`/`\endPRT` should begin their own line, as should `\selectVersion`. The markup `\uses` should follow `\ID` or `\PRT`, and there should be nothing to the right of the arguments of `\uses`.[4]

► When building your DB documents, it is not necessary that you provide solutions to the questions. Unless you are in the habit of publishing the solutions, no solution is really needed. If you don't have to provide solutions, it's much easier to build a large number of questions, with variations.

However, for problems that need vertical space for the student to respond to the question, you do need to include a `solution` environment, like so,

[4] The markup `\uses` and its arguments are removed by the scanning JS. This is the reason there should be nothing to the right of the arguments.

```
\ID{atb040506073421}
\begin{problem}[4]
Solve this problem in the space below.
\begin{solution}[2in]
\end{solution}
\end{problem}
\endID
```

4. The eqexam Package

This chapter is a subset of the documentation for the eqexam package. If this package is not already installed, do so now; see your LaTeX package manager, if there is one, or go to www.ctan.org/pkg/eqexam to obtain the download.

In the documentation that follows, I've removed all topics not relevant to building an eqexam DB document and building a paper-based eqexam document. The eqexam package has certain "online" features, see the full documentation of eqexam for details.

☛ Special comments relevant to @EASE are marked by the icon '☛' in the left margin.

1. Introduction

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

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

☛ The eqexam package is the format of the @EASE DB documents, as well as the final format of the exam document produced by @EASE.

1.1. Required and Optional Packages

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

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

Additionally, the following packages may be used depending on the options chosen. These options and packages listed below are not normally used with @EASE.

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

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

☛ I've found that the pdf option is most useful when it comes to publishing a essentially paper document created by eqexam on the Internet. The other options, links, online and email, are not normally needed for a document created from @EASE.

2. Building an Exam

In this section, we outline the steps to create an exam using the eqexam package, consult the sample exams for additional examples. For users of @EASE, the creation of the exam is automated; however, knowledge of the structure of an eqexam is necessary to build your own DB document system.

2.1. The Preamble

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

```
\documentclass{article}
\usepackage[⟨options⟩]{eqexam}
```

The ⟨*options*⟩ are discussed in section 3, page 52. Next comes the *Exam Identification Information*:

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

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

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

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

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

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

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

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

@EASE builds your LaTeX document for you, these key-value pairs are automatically inserted into your exam document, and are populated by the information you entered in the `ease.pdf` control document. Once the document is assembled, you are free to edit the file directly.

2.2. The `exam` Environment

An exam is contained within the `exam` environment.

One of the things that I do in my courses, especially for the final exam, is to have a two-part exam. Typically, the first part is worth 100 points and covers the new material not already tested; the second part is usually a 50 point review. I grade these two parts separately and record them separately. Therefore, an `eqexam` test may contain one or more `exam` environments.[1]

After the preamble, we then say

```
\begin{document}

\maketitle

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

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

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

[1] Remember, this was originally a personal package, meant to suit my own needs.

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

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

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

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

```
\instructionsColor{blue}
```

The above is the default definition.

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

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

► **Important:** You must `latex` your document *three times* to be sure all points have been properly calculated.

☛ @EASE does not support the creation of a multiple part exam. Should you wish to create such an exam, create two exams, one for Part 1 and one for Part 2, using @EASE, then merge the two files together.

2.3. The `problem` and `problem*` Environments

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

- **problem**

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

The syntax for problem is

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

The environment takes two optional arguments. The first argument, ⟨*num*⟩ is the number of points for this problem, for example, to have a 5 point question, begin the environment with \begin{problem}[5]. If we say \begin{problem}, then this problem has no points associated with it.

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

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

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

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

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

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

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

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

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

► See the section 'eqexam Options' on page 52 for more details on the `nosolutions` and `solutionsafter` options.

- **`problem*`**

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

The syntax is

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

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

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

\end{parts}
\end{problem*}
```

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

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

⟨***num***⟩**`ea`** When you specify a number followed by '`ea`' (which is short for each). Thus, '`[5ea]`' signifies that each part of this problem has weight of 5 points.

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

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

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

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

\end{parts}
\end{problem*}
```

This defines a 7 point problem.

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

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

This is what [\Do⟨*num*⟩] does. When you specify \Do3, then only the points of 3 of the problems are added into the exam total. This second parameter is only checked if the first parameter is [⟨*num*⟩ea]. \begin{problem*}[5ea][\Do3] creates a 15 point question. By the way, this assumes there are 3 or more parts to this question.

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

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

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

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

- **Page Breaking**

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

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

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

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

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

```
\nbaselineskip{8}
```

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

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

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

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

1. **Moving:** Rearrange the order of the questions, if a problem can't fit entirely on a page, you can exchange move a shorter problem to that place, and move the longer problem to another page.

2. **Tweaking:** Modify the space defined by the `solutions` environment to fit a problem on the page that is below it.

3. **Placing work on back:** Using the `\OnBackOfPage` command, page 67, you can direct the student to answer the question on the back of another page, and thus, little space is needed to follow that question.

4. **Working on separate sheets:** Of course, for some types of exams, the exam just contains the questions, the students answer the questions on separate sheets of paper. For this, you can use the `nospacetowork` option.

2.4. Special Constructs and other Gizmos

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

2.5. Fill-in Questions

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

- **Short Fill-in Questions**

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

`\fillin[u|b]{⟨width⟩}{⟨answer⟩}`

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

▶ An example of `\fillin`.

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

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

- **True/False Questions**

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

`\TF{answer}`

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

`\TF[⟨width⟩]{⟨answer⟩}`

when ⟨*width*⟩ is not specified, `\defaultTFwidth` is used (and this value can be redefined).

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

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

or this one…

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

The first alignment is the default, to get the second alignment, you need to set the value of `\fillinWidth` to the common width value of the `\TF` fields. For example,

```
\fillinWidth\defaultTFwidth
```

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

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

\fillinWidth\defaultTFwidth

\begin{parts}

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

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

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

▶ Just use `\fillinWidth` as illustrated in the above example.

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

- **Long Fill-in Questions**

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

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

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

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

2.6. Multiple Choice

For multiple choice questions, we use the `answers` environment. If the `online` or `email` option is taken, the choices are made into radio button fields so that *only one alternative* can be chosen. When multiple selections are permitted, the `manswers` environment can be used, see 'Multiple Selection' on page 48.

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

Note: No solutions are given for this problem.

Comments concerning the multiple choice questions, and their environments.

- Because the labels and values of the alternatives are based on the alphabet, the number of alternatives is restricted to twenty-six.
- The `answers` environment is borrowed from `exerquiz`, operates the same way. The one argument is the number of columns to be used in displaying the alternative answers. If the number of columns is 1, a `list` environment is used, otherwise a `tabular` environment is used.

  In the example above, we specify 6 columns for the first item (question) and 1 column for the second.
- The `\Ans` macro is used to designate which alternative is the correct answer, an argument of 1 for correct and 0 for not correct.
- The list of alternatives is enclosed by the `\bChoices` and `\eChoices` pair. There must be one alternative per line, the pair `\bChoices` and `\eChoices` also are placed on a line by themselves—as shown in the example above.

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

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

2.7. Multiple Selection

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

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

```
\begin{problem}[5]
Which of the following are primary colors?
\begin{manswers}{6} % specify tabular any with 6 columns
    \bChoices
```

```
        \Ans1 Blue\eAns
        \Ans0 Green\eAns
        \Ans1 Yellow\eAns
        \Ans0 Orange\eAns
        \Ans1 Red\eAns
    \eChoices
\end{manswers}
\begin{solution}
Yes, red, blue and yellow are primary colors.
\end{solution}
\end{problem}
```

You can use the \bChoices/\eChoices pair to specify the alternatives, or you can use the standard tabular notation. As with answers an argument of 1 specifies a list environment. See 'Multiple Choice' on page 47 for more examples on the use of the \bChoices/\eChoices pair.

2.8. Gizmos and Gadgets

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

- **The workarea Environment**

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

```
\begin{workarea}[⟨width⟩]{⟨depth⟩}
...
Material that will overwrite the solutions vertical space.
...
\end{workarea}
```

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

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

\begin{solution}[2in]
This is the solution, let's hope it's correct, or I would be
```

```
embarrassed to no end.
\end{solution}

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

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

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

- **The `\placeAtxy` Command**

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

```
\placeAtxy{⟨x_dim⟩}{⟨y_dim⟩}{⟨content⟩}
```

The first two arguments are the x and y coordinates (with dimensions) of the placement of the ⟨*content*⟩. If this comment is placed below the `solutions` environment, then the origin is the lower left corner of the solutions box.

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

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

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

- **The `splitsolution` Environment**

I developed this environment to solve a problem with the `online` and `email` options. The white space created by the `solutions` environment is converted into text fields (PDF form fields). If the `workarea` environment or the `\placeAtxy` command is used to place content on the white space, the student will be in the position of having to type on top of this content. (See the demo file `test01.tex` of the `eqexam` distribution[2] for an illustration of this.)

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

▶ Consider the following example.

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

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

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

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

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

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

[2] http://www.math.uakron.edu/~dpstory/eqexam.html

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

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

which can be redefined.

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

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

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

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

Here, the graphic appears on the left.

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

3. eqexam Options

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

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

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

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

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

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

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

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

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

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

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

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

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

pointsonboth Points are displayed in both margins.

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

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

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

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

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

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

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

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

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

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

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

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

`myconfig` If this option is taken, `eqexam` looks for the configuration file `eqexam.cfg`. This configuration file is input at the end of the package, and can be used to redefine, for language localization purposes, any of the (text) macros described in this manual. See the section 'Customizations' on page 56 for a partial listing of macros that can be redefined and placed in `eqexam.cfg`.

`myconfigi`...**`myconfigvi`** Six additional options for inputting a configuration file. If you take one of these options, `eqexam` inputs the corresponding configuration file `eqexami.cfg`...`eqexamvi.cfg`.

`cfg` Syntax: `cfg=`⟨*basename*⟩. If this option is taken, `eqexam` looks for a file named ⟨*basename*⟩`.cfg` and is input.

For one of my recent classes, I wrote many standard handouts documents: first day handout, assignment documents, homework assignments, review documents, test documents, and in-class notes. Each document-type had its own `eqexam` format (configuration file, `eqexami.cfg`…`eqexamiv.cfg`. It got confusing to keep track of all these configuration files. At which point I decided to add a *named* configuration scheme. If you use the key `cfg` in the option list `cfg=firstday`, `eqexam` will look for a file named `firstday.cfg`

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

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

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

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

► When any one of the two options above are taken, a driver needs to be specified as well, the choices are…

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

`dvipsone` For users of the Y&YTEX System, such as myself.

`pdftex` For users of pdflatex application.

`xetex` For users of xelatex application.

These options are passed on to hyperref and to eforms[3] for the proper creation of links and form fields.

☛ For @EASE, it is assumed that you have **Acrobat Professional 7.0** or later, and are using the **Adobe Distiller** to create your PDF; therefore, the dvipsone and dvips are your real choices.

[3] A component of AcroTEX Bundle.

3.1. Configuration Files

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

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

```
\ExecuteOptions{dvips}
```

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

The second configuration file, eqexam.cfg, is input at the end of the package, provided the document author takes the myconfig option. Use this file to redefine some of the commands described in 'Customizations' on page 56, and elsewhere, to customize eqexam. An obvious use for this is to have a language customization of the package, input through eqexam.cfg.

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

4. Bells, Whistles and other Customizations

4.1. Customizations

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

- **Course Info Commands**

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

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

```
\eqexamName[⟨field appearance⟩]{⟨width⟩}
```

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

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

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

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

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

```
\newcommand\eqSID[⟨field appearance⟩]{⟨width⟩}
```

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

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

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

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

```
\examSIDLabel{SID:}
```

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

```
\newcommand\eqEmail[⟨field appearance⟩]{⟨width⟩}
```

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

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

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

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

- **Changing the Title and Cover Page**

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

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

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

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

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

Command elements that are appropriate to the redefinition are \maketitledesign are…

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

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

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

\eq@ExamName This is the name field for the student to enter her/his name, as defined by default or redefined by \eqexamName, see 'Course Info Commands' on page 56.

\eq@SID This is the student ID field for the student to enter her/his ID, as defined by default, or redefined by the command \eqSID, see 'Course Info Commands' on page 56.

\eq@Email This is the email field for the student to enter her/his email address, as defined by default, or redefined by the command \eqEmail, see 'Course Info Commands' on page 56.

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

► \eqexcoverpagedesign. When the \coverpage option is taken, a default cover page appears unless it is redefined. eqexam provides the command \eqexcoverpagedesign to design your own cover page. The default cover page uses the

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

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

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

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

\eq@ExamName This is the name field for the student to enter her/his name, as defined by default or redefined by \eqexamName, see 'Course Info Commands' on page 56.

\eq@SID This is the student ID field for the student to enter her/his ID, as defined by default, or redefined by the command \eqSID, see 'Course Info Commands' on page 56.

\eq@Email This is the email field for the student to enter her/his email address, as defined by default, or redefined by the command \eqEmail, see 'Course Info Commands' on page 56.

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

- **Changing the Running Headers**

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

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

1. \lhead{⟨*text*⟩}

   Changes the left header text of the running header. This command defines an internal macro \eq@lhead that actually contains the text. The default is

   ```
   \lhead{\shortwebsubject/\shortwebtitle}
   ```

2. \chead{⟨*text*⟩}

   Changes the center header text of the running header. This command defines an internal macro \eq@chead that actually contains the text. The default is

   ```
   \chead{-- Page \arabic{page}\space of \eq@ExamLastPage\space--}
   ```

3. \rhead{⟨*text*⟩}

   Changes the right header text of the running header. This command defines an internal macro \eq@rhead that actually contains the text. The default is

   ```
   \rhead{\eq@ExamName}
   ```

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

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

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

1. `\lheadSol{`⟨*text*⟩`}`

   Changes the left header text of the running header. This command defines an internal macro `\eq@lheadSol` that actually contains the text. The default is

   ```
   \lheadSol{\shortwebsubject/\shortwebtitle}
   ```

2. `\cheadSol{`⟨*text*⟩`}`

   Changes the center header text of the running header. This command defines an internal macro `\eq@cheadSol` that actually contains the text. The default is

   ```
   \cheadSol{-- Page \arabic{page} of \eq@ExamLastPage\space--}
   ```

3. `\rheadSol{`⟨*text*⟩`}`

   Changes the right header text of the running header. This command defines an internal macro `\eq@rheadSol` that actually contains the text. The default is

   ```
   \rheadSol{SOLUTIONS}
   ```

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

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

• **Localization of Strings**

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

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

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

- `\ptsLabel`: Label for indicating the points of a problem, the default is

  ```
  \ptsLabel{pts}
  ```

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

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

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

  ```
  \pointsLabel{points}
  ```

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

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

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

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

See 'The Point and Totals Boxes' on page 65 and 'Course Info Commands' on page 56 for additional details on these and commands useful for laying out the standard text of an `eqexam` document.

4.2. Creating Multiple Versions of Exam

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

There are several convenience macros to help create exams with multiple variations. We present these macros in a series of steps for setting up a multiple version exam.

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

```
\examNum{⟨num⟩}
```

where ⟨*num*⟩ is the number to be associated with the exam (test, assignment) under construction.

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

► **Declare the number of versions:**
In the preamble, declare the number of versions for this document using `\numVersions`:

```
\numVersions{⟨nat_num⟩}
```

for example, `\numVersions{3}` states that this exam has three versions.

► **Declare which version:**
Declare which version of the exam this is for. The syntax is

```
\forVersion{⟨letter⟩}
```

For example, if you say `\forVersion{b}`, you are declaring that you want to build the exam for version 'B'. The first argument is a letter and is case insensitive, that is, typing `\forVersion{B}` or `\forVersion{b}` is the same.

When this command is executed, it defines a series of 26 commands and 26 environments: Commands `\vA`, `\vB`, `\vC`, …`\vZ`, and Environments `verA`, `verB`, `verC`, …`verZ`.

Each of the commands takes one argument. When you say `\forVersion{B}`, all the commands gobble up their arguments, all except `\vB`, which expands to its argument. All the environments become comment environments, all except `verB`, which does nothing, so the text enclosed within the environment is typeset.

For example,

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

When this is LaTeXed, the following text is typeset:

Solve the equation for y:

$$5y + 2 = 4$$

This is version 'B' as declared by `\forVersion{b}`.

► **Declare version labeling:**
Each version of the exam needs its own labeling describing the version, for example, "Test #2—Version A" or "Exam #3-Section 002". This is accomplished by the two commands `\longTitleText` and `\shortTitleText`:

```
\longTitleText
    {⟨Text1⟩}
    {⟨Text2⟩}
    ...
    {⟨Textn⟩}
\endlongTitleText

\shortTitleText
    {⟨Text1⟩}
    {⟨Text2⟩}
    ...
    {⟨Textn⟩}
\endshortTitleText
```

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

When the document is LaTeXed, the `\longTitleText` and `\shortTitleText` read their arguments and pick out the one declared in the `\forVersion` command. These commands then define two commands `\Exam` and `\sExam`; the former is the value of the long title, and the latter is the value of the short title. The two commands `\Exam` and `\sExam` are used in the `\title` command.

► **Declare Document Information:**
Finally, declare the document information needed for the exam.

```
\subject[⟨short subject⟩]{⟨subject⟩}
\title[\sExam]{\Exam}
\author{⟨instructor⟩}
\university
{%
    ⟨University/Institution⟩
}
\date{\thisterm\space\the\year} % E.g., Fall 2015
\duedate{⟨date⟩}                % actual date of the test
\keywords{\Exam, administered \theduedate}
```

Notice the use of the `\Exam` and `\sExam` commands in the arguments of `\title`.

OK? Got it? Here is an example of setting up the multiple version exam:

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

\examNum{1}
```

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

\subject[C3]{Calculus III}
\title[\sExam]{\Exam}
\author{Dr.\ D. P. Story}
\university
{%
            THE UNIVERSITY OF AKRON\\
    Department of Theoretical and Applied Mathematics
}
\date{\thisterm\space\the\year} % Fall 2015
\duedate{09/26/05}              % actual date of the test
% If you convert to pdf using a pdf links, online, email
% option, this will appear in the  keywords field of the
% document info dialog.
\keywords{\Exam, administered \theduedate}

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

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

```
\selectVersion{⟨num⟩}{⟨total_versions⟩}
```

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

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

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

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

4.3. The Point and Totals Boxes

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

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

```
\newcommand\leftmarginPtsTxt[1]{\small$#1^{\text{\eq@ptsLabel}}$}
```

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

```
\newcommand\leftmarginPtsEaTxt[1]{%
    \small$#1_{\text{\eq@eachLabel}}^{\text{\eq@ptsLabel}}$}
```

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

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

```
\newcommand\marginpointsboxtext[2]{\small$#1\,\text{\eq@ptsLabel}$}
```

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

► Points specified by the `\PTs` command. This text is defined by `\itemPTsTxt` as follows. See the paragraphs on '`problem*`' on page 42 for a discussion of the use of `\PTs`.

```
\newcommand\itemPTsTxt[1]{$#1\,\text{\eq@ptsLabel}$}
```

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

```
\newcommand\totalsboxtext{%
    \small$\theeqpointsthispage\,\text{\eq@ptsLabel}$}
```

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

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

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

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

4.4. The eqComments Environment

In addition to the `instructions` environment, see 'The exam Environment' on page 39, should you want to insert additional instructions from within the body or the exam, use the `eqComments` environment. The `eqComments` environment has one optional argument, a formatted heading for the comments you want to make. For example,

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

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

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

```
\eqCommentsColor{blue}
```

The above is the default definition.

4.5. The \OnBackOfPage Command

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

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

For this, I use the \OnBackOfPage command

\OnBackOfPage[⟨*optional text*⟩]

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

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

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

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

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

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

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

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

```
\newcommand\bopCoverPageText{\spacethe cover page}
```

We could change this as follows,

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

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

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

4.6. The \pushProblem and \popProblem Commands

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

In the example below, we have our parts in a multicols environment, we \pushProblem, close multicols, \popProblem and continue with the multi-parts in a single column.

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

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

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

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

    \item\PTs{4} Do this harder problem.
    \begin{solution}[.5in]\end{solution}
\end{parts}
\end{multicols}
\end{problem*}
```

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

☛ In order to get the correct formatting, the `multicols` environment must begin before the `parts` environment.