



Using the Exam Builder

Contents

In Version 5

1. Introducing the Exam Builder

- Checking the Hardware and Software Requirements
- Installing the Exam Builder
- Launching the Exam Builder
- Learning the Exam Builder

2. Understanding How the Exam Builder Operates

- Creating an Exam Template
- Adding a Main Setup Section
- Asking Questions
- Creating the .qiz File
- Taking and Grading the Quiz
- Detecting Errors

3. Creating Algorithmic Variations

- Changing the Order of the Questions
- Changing the Order of the Answers
- Using Question Variants
- Varying the Questions Themselves
 - Using Random Variables and Formulas
 - Using Conditions
 - Developing Choices
- Using Functions
- Working with Units
- Working with Plots

4. Enhancing Exams

- Including Answers and Detailed Solutions
- Creating Questions with Parts
- Creating Multiple-Select Questions
- Creating Questions To Be Graded Immediately
- Creating Short-Answer Questions
- Creating Exam Parts

5. Using Additional Algorithmic Generation Features

Using Definitions Created with := and =

Using Random Number Functions: rand and randmat

Using Conditions

Using Formulas

Using the Seed Keyword Paragraph

Building Questions with Algorithms

6. Delivering Course Materials

Administering Exams from a LAN or Server

Generating Printed Exams

7. Exam Builder Reference Materials

Keywords

Keyword Headings

Keyword Paragraphs

Keyword Lists

Algorithmic Generation Reference

Exam Builder Command Line Interface

Troubleshooting



In Version 5

- The exam builder can operate on a LAN such as a computer lab. Student results can be saved in a database on a computer connected to the LAN. See Chapter 6 of this help for more information.
- Installation for web use is simpler. The `verf.exe` and `tcigrade.exe` functions have been merged into the single Exam Builder executable.
- The Exam Builder is available on its own toolbar and from the Tools menu:

View Quiz



Launch Exam Builder

If you open a quiz template and click the View Quiz button, the Exam Builder runs on the template and loads the resulting quiz. It is as if you saved the template as a `.qiz` file and then opened that `qiz` file.

The Launch Exam Builder button brings up the standalone Exam Builder application.

- The standalone Exam Builder application works from a tabbed dialog.
 - The Property Page tab contains the functionality of the earlier version of the Exam Builder. It includes a new feature that automatically assigns default names to Out, Key, and Answers files associated with a template. The feature is available through the button labeled **Use default filenames**. Set the template file name and then click this button to assign default names to the other files.
 - The Database tab is used to select an `.mdb` database file and convert it to a comma-separated value (`.csv`) format, which is a plain text format that can be read by Microsoft Excel and other applications. The file generated has the same name as the original `.mdb` file but has a `.csv` extension.
- The new **OutputTap** keyword establishes a directory in which to save every quiz generated from a given template. To save the quizzes automatically, add the keyword to the template and give it a directory path. The file name for each quiz is randomly generated and displayed at the top of the quiz. This feature is useful for debugging quizzes. If you see that a quiz didn't generated properly, you can inspect the saved copy and, if you need assistance, send it to technical support. This feature can also be used to help track student activity, because it saves copies of the quizzes they take.

Chapter 1

Introducing the Exam Builder

Welcome to the Exam Builder, a set of utilities that helps you manage course materials such as exams, quizzes, exercise sets, and tutorials in **Scientific WorkPlace (SNB)** and **Scientific Notebook (SNB)**. With the Exam Builder you can

- **Create course materials easily** by using algorithmic generation
- **Save time** by having materials delivered online and graded automatically
- **Help your students learn** by increasing their opportunities for practice and giving them instant feedback

Checking the Hardware and Software Requirements

The requirements for the Exam Builder are the same as the requirements for your software:

System Requirements	SWP	SNB
IBM-compatible PC (486 or higher)	•	•
Available disk space	70–250 MB*	15–150 MB*
Windows XP, 2000, NT, Me, 98, or 95**	•	•
CD-ROM drive	•	•
Internet Explorer 3.01 or higher	•	•
Microsoft Access 97 or 2000	•	•

*Depending on the type of hard drive and installation options

**Windows Me, 98, and 95 require Windows Multilanguage Support

To take full advantage of the Exam Builder, you also need a local network or an account with an Internet service provider and an email program.

Installing the Exam Builder

The Exam Builder is installed on your computer when you install **SWP** or **SNB**; no special installation process is required. The installation process doesn't supply a desktop icon for the Exam Builder, but you can create one (see your Windows documentation for instructions). Usually, the Exam Builder is called automatically any time you open a file with the extension `.qiz`. You can also run the Exam Builder as a separate application from the **Tools** menu, the Exam Builder toolbar, or directly from a DOS prompt.

Although no special installation process is required to use the Exam Builder to create and test course materials, you must install the software on your LAN or your Windows NT server if you want to administer quizzes online and maintain student test information in a database. For full operation of the Exam Builder, your LAN or Windows NT server must also contain a copy of Microsoft Access 97 or 2000. Access isn't required on every computer.

The program CD includes a directory called EBWeb, which contains several files required for recording student test activity. Follow the instructions below to place the directory on a machine accessible to all the computers in your lab or on your Windows NT server. Regardless of where you place the EBWeb directory, it must be accessible to everyone taking your quizzes; they must be able to read, write, and execute files in the directory. Please locate this directory on your CD before continuing.

Installing the Exam Builder on a LAN or Server involves these steps:

1. Placing the EBWeb directory on the LAN or Server.
2. Modifying the switcher.tex file to reflect the location of the EBWeb directory.
3. Testing switcher.tex and the database.

Note: To run the Exam Builder successfully on your server, you may need significant knowledge of web server operation along with assistance from your system administrator. Installing on an NT Server involves these important differences:

- You must make sure that the EBWeb directory and all subdirectories have read, write, and execute permissions for arbitrary users.
- You must make sure that you don't have security features that prevent Exam Builder documents from going in or out of your server.

Step 1. Installing the EBWeb directory

► To install the EBWeb directory on a LAN

1. Use the Windows Explorer to copy the EBWeb directory to any shared drive on the LAN.

Let's assume that you place EBWeb on the C: drive of a computer named Pluto that is connected to your lab network. When you need to refer to this directory you'll use a path such as \\Pluto\C\EBWeb\...

► To install the EBWeb directory on an NT server

You must have administrative privileges and a knowledge of setting permissions for server directories and users. Contact your system administrator if necessary.

1. Copy the EBWeb directory to a directory on the NT server
We recommend the cgi-bin directory.

Step 2. Modifying the switcher.tex file to reflect the location of the EBWeb directory

The file ...EBWeb\Samples\switcher.tex lets the student enter a name and password and select a quiz. Click here to see the file as it will appear to a student.

Before this file will work correctly, you must change it to reflect the location of the EBWeb directory. Switcher.tex is a read-only file; be sure to return it to read-only status after you change the file.

► To modify switcher.tex

1. Make sure switcher.tex is read/write. Normally it is read-only.
2. Use an ASCII editor to open ...EBWeb\Samples\switcher.tex on your LAN or Server.
3. Find this line near the top of the file:

```
%TCIDATA{<FORM METHOD="POST" ACTION="//whetu/D/EBWeb/bin/eb.exe">}
```
4. If you're working on a LAN, change the path following the ACTION= to reflect the location of EBWeb in your installation.
Note the forward slashes.
5. If you're working on a Server, change the path following the ACTION= to give the URL for the EBWeb location on your web server.

The line might read something like this:

```
%TCIDATA{<FORM METHOD="POST"
ACTION="http://www.hipnt/cgi-bin/EBWeb/bin/eb.exe">}
```

Again, note the forward slashes.

6. Save the file and make it read-only.

Step 3. Testing switcher.tex

Now you should be able to open switcher.tex in **SWP** or **SNB** from any computer on your LAN and follow the instructions there to select and take several quizzes. The Exam Builder records your scores in the database ...EBWeb\Samples\databases\samples.mdb on your LAN or Server. The database contains three tables:

Table	Content
Students	Lists each student's ID, name, and password
Quizzes	Lists the quizzes you want tracked in this database
Activity	Lists who took what quiz and their score

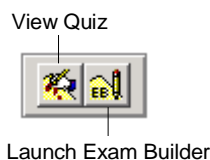
► To examine scores in the database

1. Open ...**EBWeb\Samples\switcher.tex** on your LAN or Server and take several quizzes.
2. In Microsoft Access, open the ...**EBWeb\Samples\databases\samples.mdb** database.
If you have Access 2000, the program may ask whether you want to convert the database to the format expected by the Exam Builder. Choose **No** for now.
3. Open the Activity table to find your scores.

The Database tab in the Exam Builder dialog box has a button with which you can convert a database to a file of the same name but with a comma-separated value (.csv) format, which is a plain text format that can be read by Microsoft Excel and other applications. You can use Excel to view your scores but not to edit the database.

Launching the Exam Builder

You can run the Exam Builder as a separate application directly from a DOS prompt or in **SWP** or **SNB** from the **Tools** menu or the Exam Builder toolbar:



The Exam Builder application works from a tabbed dialog.

- The Property Page tab contains the functionality of the earlier version of the Exam Builder. It includes a new feature that automatically assigns default names to Out, Key, and Answers files associated with a template. The feature is available through the button labeled **Use default filenames**. Set the template file name and then click this button to assign default names to the other files.
- The Database tab is used to select an .mdb database file and convert it to a comma-separated value (.csv) format, which is a plain text format that can be read by Microsoft Excel and other applications. The file generated has the same name as the original .mdb file but has a .csv extension.

Learning the Exam Builder

Before you start working with the Exam Builder, you should be familiar with the basic operation of **SWP** or **SNB**.

Getting Started with Scientific WorkPlace, Scientific Word, and Scientific Notebook and *Doing Mathematics with Scientific WorkPlace and Scientific Notebook*, both available online, are guides to using the software to create documents and perform computations. You can access additional information from the online Help.

Start learning about the Exam Builder with the tutorials in the following chapters. We begin by explaining how the Exam Builder operates and by constructing a basic exam template. We then add algorithmic variations to the exam and describe how to enhance exams and how to use additional algorithmic generation features. Finally, we discuss how to deliver course materials and record scores. Reference materials at the end of the manual provide additional information.

Throughout the tutorials you will find occasional links to **milestones**. Each milestone is a file that shows what your templates and quizzes should look like at a particular point in the tutorials. Use the milestones

to check your work as you complete the tutorials.

After you work through the tutorials, we suggest you study the many Exam Builder examples provided. Look over the examples in the Quizzes subdirectory of your **SWP** or **SNB** installation. Many people have already used Exam Builder to create and deliver course materials. You may be able to modify their work to suit your needs. Some materials are online; check the MacKichan Software home page for links.



Chapter 2

Understanding How the Exam Builder Operates

This chapter explains several key Exam Builder concepts:

- An Exam Builder **template** is an ordinary **SWP** or **SNB** document with a **.tex** extension.
- A template is organized into divisions, or sections, each indicated by a **keyword heading**.
- A **keyword** is a word recognized and interpreted by the Exam Builder.
- A template should consist of an identifying keyword heading, explanatory **text**, a **Main Setup Section** consisting of **keyword paragraphs**, and one or more **questions**. A template can also include comments.
- After a template has been saved with a **.qiz** extension, the program can use the Exam Builder to generate **instances** of the template. These instances are what the student actually sees.
- The questions in an instance of the template can be **graded** by the software and the results returned immediately to the student.

In this chapter, you'll use these concepts to create and test a simple Exam Builder template. As you work, you can check your exam against the milestones that appear throughout the chapter.

The chapter presents these keywords: Exam, Comment, Text, Title, Question, Chices, Setup, Submit, and Statement.


Creating an Exam Template

You create Exam Builder templates using **SWP** or **SNB**. After you've gone through the entire Exam Builder process a time or two, you'll probably want to start your template by using the time-honored method of copying an existing template and modifying it. But let's do it from scratch once first. In this tutorial we'll create a set of factoring exercises for an elementary algebra course. The students will use the exercises as an online drill.

Step 1: Start a New Exam Builder Template

A template is nothing but an **SWP** or **SNB** document with some special content.

► To start a new template

1. In **SWP** or **SNB**, click  or, from the **File** menu, choose **New**.
2. Select the shell directory and shell file you want to use. For this tutorial, select the Scientific Notebook shell directory in **SWP** or the General shell directory in **SNB** and then select the Blank Document shell file.
Although you can use any document shell to create an Exam Builder template, the program includes several shells specifically designed for creating exams. In the future, try the Exam with instructions and Ranger College shells in the Scientific Notebook shell directory in **SWP** or the Exams and Syllabi shell directory in **SNB**.
3. Choose **OK**.
4. From the **File** menu, choose **Save As**.
5. Type a name for the file. Let's call it **firstquiz.tex**.
6. Choose **OK**.

You should now have a blank document titled **firstquiz.tex** open in the program window.

Step 2: Add an Exam Keyword Heading

Templates are divided into major divisions, or sections, that begin with **keyword headings**—section

headings that contain keywords recognized by the Exam Builder. For a complete list of the keywords, see the Keyword Reference at the end of the tutorial. Our template begins with a keyword heading identifying it as an exam.

Note that although the instructions below call for a Heading 1 tag, you can use any section tag to contain a keyword heading. For the Exam Builder's purposes, all section tags—Heading 1, Chapter, Section, and so forth—are all the same. The Exam Builder only looks at the enclosed keyword. Here we follow convention, using the visually larger tags such as Heading 1 or Chapter for the more important template divisions and using the smaller headings for the lesser divisions.

► To add a keyword heading

1. If the insertion point isn't already at the beginning of a paragraph, press **Enter** to begin a new paragraph.
2. Apply a Heading 1 tag using the Section/Body Tag popup list, the function keys, or the **Apply** command on the **Tag** menu.
3. Type the keyword **Exam**.
4. Press **Enter**.

Step 3: Add a Comment

Next we'll add a comment explaining the purpose of the template. A comment is started with a keyword heading using the keyword *Comment*. All text that falls between this heading and the next keyword heading is a comment intended solely for notes to yourself. Comments are always optional, but their occasional use may help you understand your own work.

► To add a comment

1. Apply a Heading 3 tag.
2. Type the keyword **Comment** and then press **Enter**.
3. Type your comment.
For our purposes, type "This is the tutorial firstquiz.tex."

Step 4: Add an Exam Header

An explanatory header that appears at the top of the quiz is useful for the student. For example, you might want the exam to start something like this:

Math 101 Drill 1

Instructions: Practice this exercise set until you can answer all questions correctly in less than 1 minute.

Use a *Text* keyword to enter the header. The Text keyword does two things. First, it ends the comment that you wrote in the previous step. Second, it begins a section in which you can enter any text that you want to appear explicitly on the exam.

► To add an exam header

1. Apply a Heading 3 tag.
2. Type the keyword **Text** and then press **Enter**.
3. Type your header.

For the purposes of this tutorial, you can copy the header shown above and paste it into your template.

★ **Check your work**

To check your work, click this link to the tutorial's first milestone.

Adding a Main Setup Section

The Main Setup Section is where you place all the options that govern the overall generation of instances from the template.

Step 5: Start the Main Setup Section

- ▶ To start a Setup section
 1. Apply a Heading 1 tag.
 2. Type the keyword **Setup** and then press **Enter**.

Step 6: Add a Title Paragraph and a Submit Paragraph

A Setup section contains a number of **keyword paragraphs**: paragraphs that begin with a keyword followed by a colon. For a complete list of keywords, see the Keyword Reference at the end of the tutorial.

We'll now add two keyword paragraphs to the Main Setup Section of the template. The first keyword paragraph contains the title that will appear on the window opened by the student. The second keyword paragraph contains the caption that will appear on the Submit button in the student's window. The student uses the button to submit the answers for automatic grading. We'll cover submitting and grading answers in more detail later in the tutorial.

- ▶ To add a Title paragraph and a Submit paragraph
 1. If the insertion point isn't at the beginning of a paragraph, press **Enter** to start a new paragraph.
 2. Type the keyword **Title:** (note the colon) followed by the title of the exercise.
For this tutorial, use the title **Practice Drill 1**.
 3. Press **Enter**.
 4. Type the keyword **Submit:**(note the colon) followed by the phrase you want to appear on the Submit button.
For this tutorial, type **Click to Grade**.
 5. Press **Enter**.

★ Check your work

To check your work, click this link to the tutorial's second milestone.

Asking Questions

Now we want to pose a question for the student.

Step 7: State a Question

We'll create a multiple-choice question that asks for a factorization of $x^2 - 2x + 1$.

- ▶ To state the question
 1. Create a keyword heading to start a new question:
 - a. Apply a Heading 1 tag.
 - b. Type the keyword **Question**.
 - c. Press **Enter**.
 2. Create a keyword heading to start the question statement:
 - a. Apply a Heading 3 tag.
 - b. Type the keyword **Statement**.
 - c. Press **Enter**.
 3. Type the question: **Factor** $x^2 - 2x + 1$.

Step 8: List the Possible Answers in a Choices Section

We'll list four possible answers: $(x - 1)^2$, $(x - 1)(x + 1)$, $(x - 2)(x + 1)$, and *None of these*.

► To list the possible answers

1. Create a keyword heading to start the answers:
 - a. Apply a Heading 3 tag.
 - b. Type the keyword **Choices**.
 - c. Press **Enter**.
2. Start a bulleted list: apply a Bulleted List Item tag.
3. Type the first possibility: $(x - 1)^2$.
Note: Don't press **Enter** yet.
4. From the Fragments popup list, select **Correct Choice** and then press **Enter**.
5. Type the second possibility $(x - 1)(x + 1)$ and press **Enter**.
6. Type the third possibility $(x - 2)(x + 1)$ and press **Enter**.
7. Press **F2** to close the bulleted list.
8. Start a numbered list for the final possibility: apply a Numbered List item tag.
Items in bulleted lists may be rearranged by the Exam Builder, but items in other kinds of lists remain in place. Because we want the final possibility, *None of these*, to be the last choice, we must place it in a numbered list by itself.
9. Type the final possibility, **None of these**.
Note: You can emphasize the fact that this list item is in a fixed place by revising the numbered item lead-in object and creating a custom lead-in object such as *Fixed*.

★ **Check your work**

This is a good time to check your work. Click this link to the tutorial's third milestone.

Step 9: Add More Questions

► To add more questions

1. Repeat Steps 7 and 8 to add a few more questions to your exam.
2. Save the file and continue the tutorial.

or

- If you don't want to make up your own questions, you can use the questions in the tutorial's fourth milestone:
 1. Click this link to the fourth milestone.
 2. From the **File** menu, choose **Save As**.
 3. In the **File name** box, type **firstquiz.tex** and press **Enter**.
 4. Continue the tutorial.

Creating the .qiz File

Now we have a large enough file to test. Before we can take and grade the quiz, we have to create a .qiz file from the exam template.

Step 10: Create the .qiz File

After you save the exam template with a .qiz extension, the Exam Builder can generate **instances** of the template. These instances are what the student sees.

► To create the .qiz file

1. From the **File** menu select **Save As**.
2. In the **File name** box, type **firstquiz.qiz**.
You may use any file name and directory you like. Just be sure that the file extension is `qiz`.

3. Choose **OK**.
4. Close any open windows, except this help window.

Taking and Grading the Quiz

Put on your student hat and take the quiz.

Step 11: Take the Quiz

Each time you or your student takes the quiz, the Exam Builder, running in the background, reads the .qiz file and creates an instance of the template. The program then displays a quiz document containing the header, questions, and choices you have created. After the answers are sent to the grader, the program grades the quiz and displays the results.

► To take the quiz

1. From the **File** menu choose **Open**.
2. In the **Files of Type** box, select **Quiz (*.qiz)**.
3. Select your .qiz file.

The Exam Builder creates an instance of your template and displays it in a new window. In this tutorial, the title of the window is *Practice Drill 1*. The choices for each question are each preceded by a radio button, which is a default for multiple-choice questions. At the end of the quiz is the default Submit button, a gray button labeled *Click to Grade*.

Note: If the Exam Builder detects an error in the template, it will not generate an exam instance. Instead, it will return a copy of the template marked up with bright red error messages. See the information below regarding detecting errors. If your template has an error that you can't correct, continue the tutorial by using the fourth milestone: open it and save it as **firstquiz.tex** and then repeat Steps 8, 9, and 10, above, and take the quiz again.

4. Answer each question by clicking the button for one of the possible answers.
5. When you have answered all the questions, click the **Click to Grade** button at the end of the quiz.

The Exam Builder collects your answers, sends them to the grader, and displays the results in a new window. The results include your score, the time you spent on the quiz, and, for each question, the question statement, your answer, and the correct answer.

Take the quiz several times; the questions will be the same each time. Try answering some questions correctly and some, incorrectly. Try leaving some questions unanswered. Note how the grader reports the results in each case.

Detecting Errors

The Exam Builder detects certain syntactic errors when compiling a template. If you compile a template, say by loading a .qiz file in **SWP**, and the Exam Builder detects an error, then an exam instance will not be created and loaded. Instead, a copy of the template marked up with error messages (in large bright red letters) will be created and loaded. You must correct the error before the program can create an exam instance. Click here to see what happens when you try to load a .qiz with syntax errors; here is the template. When you use a quiz created with Version 3.0, you are likely to encounter errors of the sort demonstrated in that template. because this version of the Exam Builder is less tolerant of errors than earlier versions. So when the new version indicates an error, you should fix the offending part of the template. See the Reference section for more information.

Here are some other typical errors:

1. Sometimes errors are caused by spacing accidentally inserted in keyword headings and other constructs. For example, the following heading has a vertical space (a small, green vertical arrow) just before the letter T; to see it, turn on **Invisibles** from the **View** menu.

This is a heading with a vertical space at the beginning

2. Another spacing error appears in the following mathematics, which looks correct. However, if you turn on invisibles, you see that a space between the a and the left parenthesis: $x + y = a$
($u - v$) You may have to increase screen magnification to see the space.
3. Errors occur if you create algorithmically generated questions with conditions that might not be satisfied. To catch these errors, you should set your computational engine error handling to _____ while developing a quiz. This will cause a message to be generated if a question is generated without the associated conditions being satisfied.

★ Here's another example

Click this link to try a multiple-choice quiz containing several questions that demonstrate the use of a *None of these* choice. The exam template for the quiz is Hecht-1.tex in the **Quizzes** directory of your installation; it is a good example of a complete exam template.

In the chapters ahead...

The tutorial template we've constructed so far is very basic. To create a useful template, we have more work to do and many more options to consider. In the next chapter, we'll add algorithmic variation to the quiz.



Chapter 3

Creating Algorithmic Variations

Much of the Exam Builder's power lies in **algorithmic variation**—the automatic generation of slightly differing instances of course material based on a single template. This chapter explains how you can use the Exam Builder to build algorithmic variations into your file in several ways. You can

- Randomly change, or **permute**, the order of the questions and answers on the exam.
- Provide several equivalent questions, or **variants**, for each question and randomly **select** one or more for inclusion in the instance.
- Vary the questions themselves by creating questions and answers using random number functions, conditions, and formulas.

In this chapter, we'll work with several exam templates—the one we created earlier (`firstquiz.tex`) and another that we'll create.

The chapter presents these keywords: `Permute`, `Variant`, `Condition`, `Select`, and `Variant`.

Changing the Order of the Questions

The easiest way to give the students variety of quizzes is to allow Exam Builder to vary the order of the questions.

Step 1: Tell the Exam Builder to Vary the Order of the Questions Automatically

We'll open and modify our existing exam template so that the Exam Builder varies the order of the questions in each instance of the template without any further intervention on our part.

► To vary the order of the questions automatically

1. Open the exam template **firstquiz.tex**, which you created in Chapter 2.
If you don't have the file, use the tutorial's fourth milestone.
2. Scroll to the Main Setup Section.
It currently contains two keyword paragraphs, **Title:** and **Submit:**
3. Position the insertion point after the **Submit:** paragraph.
4. Add a new keyword paragraph: type the keyword **Questions:** followed by the keyword **Permute**.
5. Save the template **firstquiz.tex**.

Step 2: Take the Quiz

Remember that you must create a `.qiz` file before you can take the quiz.

► To take the quiz

1. Create a `.qiz` file:
 - a. From the **File** menu choose **Save As**.
 - b. In the **File name** box, type **firstquiz.qiz**.
2. Close the `.qiz` file and then reopen it.
3. Take the quiz. Pay close attention to the order of the questions.
4. Repeat steps 2 and 3 several times.
Note that the order of the questions varies randomly over time.

Changing the Order of the Answers

Another easy way to vary a quiz is to let the Exam Builder vary the order of the answers.

Step 1: Tell the Exam Builder to Vary the Order of the Answers Automatically

We'll modify our existing exam template so that the Exam Builder automatically varies the order of the answers in each instance of the template.

► To vary the order of the answers automatically

1. Open the exam template **firstquiz.tex**.
2. Scroll to the Main Setup Section.
It contains the keyword paragraphs **Title:** and **Submit:** and, if you completed the previous section, a **Questions: Permute** line.
3. Delete the **Questions: Permute** line.
4. In place of the deleted line, add a new keyword paragraph: type the keyword **Choices:** followed by the keyword **Permute**.
5. Save the template **firstquiz.tex**.

Step 2: Take the Quiz

Create a .qiz file for the template and then take the quiz several times, paying close attention to the order of possible answers. Note that the order of the answers varies randomly over time.

Using Question Variants

A question may be constructed from a number of **variants**, or questions that are in some way equivalent to each other. You usually want to ask a student just one of the variants. For example, you might want to ask a standard word problem: "A car leaves town A at 12:00 and travels north at 50 mph...." A natural variant of this question is "A boat leaves port A at 2:00 and travels east at 20 mph...." With the Exam Builder you can create questions from a list of variants. Every time the Exam Builder generates an instance of the question, it selects one or more of the variants, depending on your instructions.

Step 1: Create a Question Consisting of Variants

Use the standard word problems suggested above to create several question variants, each with its own question statement and multiple-choice responses.

► To create a question consisting of variants

1. Create a new **Question** keyword heading: apply a Heading 1 tag, type the keyword **Question**, and press **Enter**.
2. If you want the Exam Builder to select more than one variant from the list of variants you provide:
 - a. Create a **Setup** keyword heading: apply a Heading 3 tag, type the keyword **Setup**, and then press **Enter**.
 - b. Create a **Select:** keyword paragraph:
 - i. Apply a Heading 3 tag and type the keyword **Select:**
 - ii. Type the number of variants you want the Exam Builder to select.
 - iii. Press **Enter**.
3. Create a **Variant** keyword heading: apply a Heading 1 tag, type the keyword **Variant** and then press **Enter**.
4. For the variant, create **Statement** and **Choices** keyword headings, together with appropriate statement and multiple-choice responses:
 - a. Apply a Heading 3 tag, type the keyword **Statement**, and press **Enter**.
 - b. Type the question.
 - c. Apply a Heading 3 tag, type the keyword **Choices**, and press **Enter**.
 - d. Create a list containing the possible answers and indicate the correct choice.
5. Repeat steps 3 and 4 for each variant. Add as many variants as needed.

6. Save the template as **myvariants.tex**.

Step 2: Take the Quiz

Create a .qiz file for the template and then take the quiz several times, paying close attention to the questions that appear. Note that they vary randomly over time.

★ Here's another example

Click this link to try the quiz history.tex, supplied in the **Quizzes** directory. The quiz is a true/false quiz that contains just one question consisting of several variants. The question setup tells the Exam Builder to select 10 of the variants for inclusion in the generated quiz. Click here to take the history quiz. If you take the quiz several times you will see not only questions that appear in different order but also completely different questions. The quiz is a complete example of the use of variants.

Varying the Questions Themselves

Now that you have an idea of how to vary the order of questions and answers, let's look at ways to vary the question statements themselves. Let's say we want a question of the form "Factor $ax^2 + bx + c$ " and we want a, b and c to vary each time the question is generated. We can take several approaches to creating the question. One is to use question variants—make a list of all your favorite problems of this type and include them as variants of a single question, as described in the previous section. Another approach is to vary the questions themselves by using **random variables and formulas**, and imposing **conditions**.

Using Random Variables and Formulas

We can vary the question "Factor $ax^2 + bx + c$ " by letting the variables a, b, c vary randomly. Let's see how to do that. First, we have to be careful about what we mean. We don't want a, b , and c to be completely arbitrary. For starters, let's say we want the variables to be natural numbers between 1 and 10.

Step 1: Use Random Variables in the Question

We'll create a different exam template to explore the use of random variables.

► To use random variables in the question

1. Open milestone 2 as a starting point.
2. Save it as **secondquiz.tex**.
3. Create a **Question** keyword heading: apply a Heading 1 tag, type the keyword **Question**, and press **Enter**.
4. Create a **Setup** keyword heading: apply a Heading 3 tag, type the keyword **Setup**, and then press **Enter**.
5. Insert the following lines of mathematical definitions:
$$a := \text{rand}(1, 10)$$
$$b := \text{rand}(1, 10)$$
$$c := \text{rand}(1, 10)$$

These lines assign a, b , and c random integer values between 1 and 10 **every time** the template is used to generate a quiz.

6. Create a **Statement** keyword heading: apply a Heading 3 tag, type the keyword **Statement**, and press **Enter**.

Step 2: Use a Formula to State the Question

Now we want to write the statement "Factor $ax^2 + bx + c$." However, if we simply enter the statement, the student will see it without variation. What we really want is for the student to see random values **substituted for the variables**. To do this we use the program's formula feature. Here are the steps.

► To state a question with a formula

1. From the **View** menu, turn on **Helper lines**.
2. Type **Factor** and press the space bar.
3. From the **Insert** menu, choose **Formula**.
The **Formula** dialog box opens.
4. In the **Formula** box, type $ax^2 + bx + c$
5. In the **Operation** box, choose **evaluate**.
6. Choose **OK**.
In the document window, the formula appears with a yellow background, like this:
Factor $ax^2 + bx + c$.
7. Save your file as **secondquiz.tex**.

★ **Check your work**

Now check your work. Click this link to Milestone 6.

Step 3: Take the Quiz

Create the **secondquiz.qiz** file and then open it and close it several times. You will see the question vary. If you haven't made your own .qiz file, here is the .qiz version for you to try. Note that the file doesn't yet contain any possible answers; we'll add them later.

Step 4: Make Sure the Problems Factor Nicely

Of course, not all questions produced randomly can be factored nicely. Ideally, we want numbers u_1, u_2, u_3, u_4 so that $ax^2 + bx + c = (u_1x + u_2)(u_3x + u_4)$. The easiest thing to do is select the u_i and then generate a, b, c .

► To make sure that the posed problems factor nicely

1. Open **secondquiz.tex**.
2. Replace the setup lines that define the variables a, b, c with the following lines:


```

 $u_1 := \text{rand}(1, 10)$ 
 $u_2 := \text{rand}(1, 10)$ 
 $u_3 := \text{rand}(1, 10)$ 
 $u_4 := \text{rand}(1, 10)$ 
 $a := u_1u_3$ 
 $b := u_1u_4 + u_2u_3$ 
 $c := u_2u_4$ 

```
3. Add the following debugging line to the question statement:
The answer is $(u_2 + xu_1)(u_4 + xu_3)$
The factors are entered in two separate formula objects.
4. Save the .tex file.

★ **Check your work**

Check your work against Milestone 7.

Step 5: Take the Quiz

Save **secondquiz.tex** as a .qiz file and then take the quiz several times. Or, click here to see the quiz in action. Now you'll get problems that factor nicely.

Using Conditions

Frequently we want to impose a condition on the random variables. For example, we might want the $\text{gcd}(u_1, u_2) \neq 1$ so that the factor $u_1x + u_2$ has an integer factor, and we may expect the student to factor the integer out as well.

To impose a condition on the random variables in a Setup section, we add a Condition keyword followed by the condition, as shown in the instructions below. Conditions can be combined with the boolean operators \wedge (and) and \vee (or). You can set up conditions on more than one line; begin each line with the keyword Condition.

Step 6: Create the Conditions

Suppose that in addition to the gcd condition, we also don't want the numbers in the problem to get too big. Let's say that the sum of the u 's should be less than 20.

- ▶ To make sure the question has an integer factor and the numbers aren't too big
 1. Open **secondquiz.tex** or, if you haven't made secondquiz, open Milestone 7.
 2. In the Setup section for the question, add these lines defining d as the gcd of u_1 and u_2 and e as the gcd of u_3 and u_4 :

$$d := \text{gcd}(u_1, u_2)$$

$$e := \text{gcd}(u_3, u_4)$$

This saves some typing, because we'll use the gcd's more than once.

3. State the condition that the sum of the u 's should be less than 20: at the end of the Setup section for the question, add this line:

$$\text{Condition: } d > 1 \wedge u_1 + u_2 + u_3 + u_4 < 20$$

We don't care whether e is greater than 1 as long as d is.

4. Change the formulas in the debugging line to show the integer factor:

$$(de)\left(\frac{1}{d}u_2 + \frac{1}{d}xu_1\right)\left(\frac{u_4}{e} + x\frac{u_3}{e}\right)$$

5. Save the template as **secondquiz.tex**.
6. Save the template as **secondquiz.qiz**.

Step 7: Take the Quiz

Take the quiz several times by closing and opening the .qiz file. You can check your work against Milestone 8.

You may notice that what you see highlighted in yellow on the screen is not exactly what you typed into the formula dialog. If you double click the middle formula above, you'll see that although it appears as $\frac{u_1}{d}x + \frac{u_2}{d}$, it really contains $(u_1/d)x + (u_2/d)$. The reason is that **SWP** and **SNB** always ask for the entered formula to be evaluated before displaying it. And removing the parentheses is a typical simplification by the computational engine. You may sometimes get very strange things from this mechanism. To see what a formula actually contains, disregard what you see on the yellow background (which may not be accurate) and open the formula object.

Developing Choices

Now we have a template that generates reasonable factoring problems. Let's develop some choices for the students.

Step 1: Create the Choices Section

Suppose we want to provide four answers: the correct answer, the answer *None of the above; the polynomial is irreducible*, and two other incorrect choices. In working the problem, one typical student mistake might be to forget to factor out the integer and get $(u_1x + u_2)(u_3x + u_4)$. Another might be to get the constant terms switched and guess $(u_1x + u_4)(u_3x + u_2)$. We'll use these as our incorrect guesses.

Now this raises the problem that these two answers might not be distinct. For example, suppose the random number generator makes $u_2 = u_4$. Then these two incorrect answers are not distinct. We can fix the problem by adding the condition that $u_2 \neq u_4$.

- ▶ To create the Choices section

1. Open **secondquiz.tex** or, if you haven't created the file, open Milestone 8.

2. Scroll to the **Condition** keyword paragraph in the Setup section for the question.
3. At the end of the section, add the condition $u_2 \neq u_4$.
4. Remove the debugging line from the question statement.
5. After the question statement, insert a Choices keyword heading:
 - a. Place the cursor at the end of the last line of the file and press **Enter** to start a new paragraph.
 - b. Apply a Heading 3 tag, type the keyword **Choices**, and press **Enter**.
6. Enter the first three choices:
 - a. Apply a Bulleted List Item tag to start a bulleted list for the choices.
 - b. Type the correct choice: $(de)\left(\frac{1}{d}u_2 + \frac{1}{d}xu_1\right)\left(\frac{u_4}{e} + x\frac{u_3}{e}\right)$.
 - c. From the Fragments popup list, select **Correct Choice** and then press **Enter**.
 - d. Type the next two choices: $(u_2 + xu_1)(u_4 + xu_3)$ and $(u_4 + xu_1)(u_2 + xu_3)$.
 - e. Press **F2** to exit the bulleted list.
7. Enter the final choice:
 - a. Apply a Numbered List Item tag.
 - b. Type the final choice: **None of the above; the polynomial is irreducible**.
 - c. Press **Enter** and then press **F2** to exit the list.
8. Save the exam template.
9. Create the .qiz file.

Step 2: Take the Quiz

Try your quiz or, if you didn't save the file, click here to take the quiz.. Check your work against Milestone 9.

Note that the conditions we impose must be relatively easy to satisfy. If after a few hundred attempts the Exam Builder can't find a random assignment to the variables that satisfies the condition, it returns undefined results. For example, here is a condition that is hard to satisfy:

$a := \text{rand}(1000000)$

$b := \text{rand}(1000000)$

Condition: $a = b$

What are the chances that two numbers selected at random from such a large specified range will be equal? This condition will not be satisfied very often. Do not use this sort of condition in your exams.

Using Functions

You can define functions in the Setup section of an exam and use them in formula objects in the same way you define and use variables. The process is sometimes useful in reducing the typing in the template. These functions are intended for the benefit of the exam writer and not for the exam taker, who will never see them. Also, do not confuse this kind of definition with the **Define** command on the **Compute** menu, which should never be used in a quiz template. Definitions added to Setup sections are not defined to the computational engine until the template is loaded by the quiz taker.

★ Here's an example

As an example, to define a function $\text{nplaces}(x, n)$ that returns x to n decimal places (provided your computational engine settings allow it), you can add the following line to your Main Setup Section.

$$\text{nplaces}(x, n) = 1.0 \cdot \lfloor 10^n x + .5 \rfloor / 10^n$$

Note the use of $=$ rather than $:=$. This line defines a function that takes two arguments and displays the first argument x to the number of decimal places specified by the second argument n . You can write questions such as "Calculate π to 3 decimal places" and then use the function to help generate the results.

See Milestone 10 for the complete example and click here to try out the quiz. Also see the next example,

which uses units in conjunction with function definitions.

Working with Units

You can use units of measure in the expressions of Exam Builder question setups, statements, and responses. For a couple of ideas on things you can do, see this template and take this quiz.

Working with Plots

Note: The snapshot files prevent your seeing the same thing the quiz taker sees. Before writing a quiz with generated plots, turn off automatic snapshot generation (from the **Tools** menu, choose **Computation Setup**, and then choose **Plot Behavior**).

You can use plots in Exam Builder question statements, choices, and solutions. As an example, suppose we want to ask a question such as "Which of the following is the graph of $y = x^2 - 5x - 6$?" We want to follow the question with some possible graphs. To include the plots, we add several lines to the question Setup section. We then randomly select two roots, a and b , for our parabola, and then define the variable $y := (x - a)(x - b)$. In the Choices section, we specify the plot.

► To include the plots

1. Create a new Question section.
2. Create a Setup section for the question.
3. Type these lines:
 $a := \text{rand}(1,4)$
 $b := \text{rand}(1,4)$
 $y := (x + a)(x - b)$
Condition: $a > b$
4. Press **Enter**.
5. Create a Statement section for the question.
6. Type the question using formulas as shown here:
Which is a graph of $y = x^2 + x(a - b) - ab$?
7. Press **Enter**.
8. Create a Choices section for the question.
9. Start a bulleted list for the choices:
 - a. Apply a Bullet List item tag.
 - b. Type y and then from the **Compute** menu, choose **Plot 2D, Rectangular**.
The computational engine plots a straight line.
 - c. Delete the y .
 - d. Repeat steps b and c with other variables and then press **Enter**.
 - e. Press **F2** to exit the list.
10. Save the exam template as a .tex file.
11. Create the .qiz file.

★ **Check your work**

Compare your .tex file to Milestone 15 and take the exam.

Note: When you plot mathematics, it is important to assign the expression being plotted to a variable (y in this example) and plot the variable. Do not try to put complex expressions directly into a plot.

Chapter 4

Enhancing Exams

Exam Builder exams can consist of much more than simple questions and answers. In this chapter, we look at enhancing exams by

- Providing detailed solutions to exam questions
- Creating questions with several parts
- Creating multiple-choice questions with more than one correct answer
- Creating questions that are graded immediately
- Creating short-answer questions
- Creating exams with several parts

The chapter presents the GradProc function and these keywords: Images, Answer, Solution, Substatement, and Part.

Including Answers and Detailed Solutions

With the Exam Builder, you can provide solutions to your exam questions that students can use online or on paper. After students submit their answers for online grading, they can immediately compare their answers with your solutions. If students take the exam on paper, you can print the answer key when you generate and print the exam, then distribute the key after the exam has been completed.

The Exam Builder provides two similar sections, each headed with keywords, for providing problem solutions: **Answer** sections and **Solution** sections. Usually, Answer sections contain brief solutions. Solution sections contain longer solutions, which usually include tutorial material that explains in detail the process of obtaining the correct answer. If you include both an Answer section and a Solution section for a question, both appear in the solutions file for the exam.

Let's add some solution materials to the tutorial exam template we've created.

► To include answers and solutions for a question

1. Open Milestone 9.
2. Scroll to the end of the Choices section and press **Enter**.
3. Create an Answer section:
 - a. Apply a Heading 3 tag to the new paragraph.
 - b. Type the keyword **Answer** and press **Enter**.
 - c. Type an answer for the question, such as this:

The answer is $(de)\left(\frac{1}{d}u_2 + \frac{1}{d}xu_1\right)\left(\frac{u_4}{e} + x\frac{u_3}{e}\right)$

Use formulas to generate the correct answer just as you did in the Choices section.

- d. Press **Enter**.
4. Create a Solution section:
 - a. Apply a Heading 3 tag to the new paragraph.
 - b. Type the keyword **Solution** and press **Enter**.
 - c. Type a solution for the question such as this:
 $(u_2 + xu_1)(u_4 + xu_3)$ **is wrong because there is a factor of d in $u_2 + xu_1$.**
Again, use formulas to generate the correct answer.
 5. Save the template as a .tex file.

★ **Check your work**

Compare your work to the Answer and Solution sections at the bottom of Milestone 11.

★ **Here's another example**

Click this link to see how Answer sections are used in the quiz history.tex, supplied in the **Quizzes** directory.

Creating Questions with Parts

Sometimes we want to create a question that has several parts. We can set up multiple-part questions with a **Substatement** section. Substatement sections appear before the Choices section for a question. Let's create a question with two parts, A and B, in an exam template.

► **To create a question with multiple parts**

1. Start a new exam template with an Exam header.
2. Create a Question section and press **Enter**.
3. If needed, insert a Setup section and press **Enter**.
4. Start a Statement section and press **Enter**.
5. Type the problem statement and press **Enter**.
For this example, type **Suppose you have a dozen Lincoln pennies**.
6. Create part A of the question.
 - a. Start a Substatement section:
 - i. Apply a Heading 3 tag to the new paragraph.
 - ii. Type the keyword **Substatement** and press **Enter**.
 - b. Type part A of the question: **Whose profile is on the coins?**
 - c. Start a Choices section.
 - d. Use a bulleted list and a numbered list to state the possible answers for part A: **Abe Lincoln, George Washington** , and **None of these**.
 - e. Press **Enter**.
7. Create part B of the question.
 - a. Start a new Substatement section and press **Enter**.
 - b. Type part B of the question: **The total value of the coins is**
 - c. Start a Choices section.
 - d. Use lists to state the possible answers for part B: **12 cents; 2 dollars; It depends on the mint, year, and condition of the coins;** and **None of these**.
8. Save the template as a .tex file.
9. Save the template as a .qiz file.

★ **Check your work**

Compare your work to Milestone 12, and take the corresponding quiz here.

Creating Multiple Select Questions

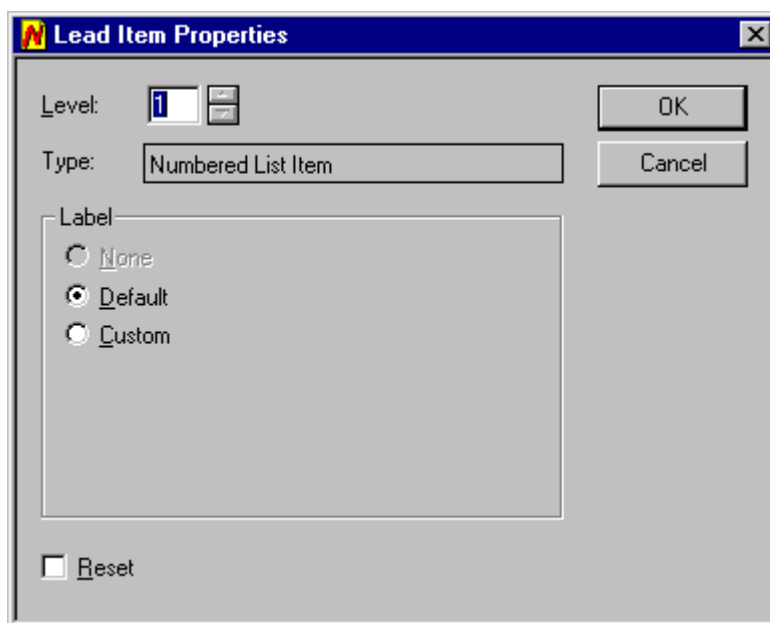
Another way to vary exams is to create questions with more than one correct answer. You can use the Exam Builder's **multiple-select questions** to present the student with a collection of choices, some correct and some not. To get full credit for the question, the student must select all the correct choices and only those choices. Multiple-select questions are a variation on multiple-choice questions. Let's use a new exam template to create a multiple-select question.

► **To create a multiple-select question**

1. Start a new exam template with an Exam header.

2. Create a Question section and press **Enter**.
3. Start a Setup section: apply a Heading 3 tag to the new paragraph, type the keyword **Setup**, and press **Enter**.
4. Create a keyword paragraph with the Choices keyword:
 - a. Type **Choices**:
 - b. Type the keyword **Check**.

This paragraph tells the Exam Builder to use check boxes rather than radio buttons when it generates the choices.
5. Start a Statement section and press **Enter**.
6. Type the problem statement.
For this example, type **Which of the following are prime?**
7. Start a Choices section and enter your list of choices.
For this example, use these possible answers: **1, 2, 3, 9, 11**.
8. For each list item, create a custom lead-in object that indicates the points you want assigned to that choice.
 - a. Double-click the lead-in object of the first list item to open the **Lead Item Properties** dialog box.



- b. Click the Custom button to open an input area.
 - c. Enter the number of points the answer is worth.
The number should be positive if the selection is correct and negative otherwise. In our example, you might enter **-1** for the first and fourth answers and **1** for the second, third, and fifth answers
9. Save the template as a .tex file.
10. Save the template as a .qiz file.

★ **Check your work**

Compare your work to Milestone 13, and click here to take the associated quiz. Take this quiz a few times with various combinations of correct and incorrect answers and observe how the Exam Builder grades questions.

★ **Here's another example**

In this example, the template appears first, followed by the quiz as seen by the student:

Question

Setup
Choices: Check, Permute, No Break

Statement
Which of these numbers are prime?

Comment
The only correct answer is 3 and it is worth 1 point. This is the maximum possible for the question. By checking all the incorrect boxes you may get -6 points.

Choices

- 1 3
- 2 12
- 2 15
- 1 111
- 1 1001

1 Which of these numbers are prime?

12 1001 3 111 15

Creating Questions To Be Graded Immediately

When you want a student to answer a question and grade it immediately, present your answers in **image-select questions**. Image-select questions are another variation on multiple-choice questions. Instead of listing each possible choice preceded by a radio button, image-select questions present the possible answers graphically using arbitrary images (.bmp files). When the student clicks an image, the Exam Builder grades the question immediately. Therefore, image-select questions are useful only for single-question quizzes, perhaps presented before the student answers further questions or progresses to a subsequent lesson of a tutorial.

The images you use in image-select questions are .bmp files that you indicate, possibly with a path. The file names can be

- Relative paths to the directory for the source
- Relative paths to the HTTP base given in the exam Setup section (see Running the Exam Builder on a Server)
- Absolute paths

For ease, we recommend putting the .bmp files in the same directory as the Exam Builder template and then listing only the file names (rather than whole path names) in the Images keyword paragraph.

Let's create an image select question in a new exam template.

► To create an image-select question

1. In a directory containing some bitmap (.bmp) images, start a new exam template. The images might be called first.bmp, next.bmp, and so on.
2. Create an Exam header.
3. Create a Setup section.
4. Start an Images: keyword paragraph:
 - a. Type **Images:**
 - b. After the colon, type a list of the names of the bitmap files that will be used to signal each possible choice.
List the names in the order you want them to appear in the generated quiz.
For example, your template may contain a line something like this:
Images: first.bmp, next.bmp,...,last.bmp
5. Create a Question section and press **Enter**.
6. Create a Statement section and type the statement of your question.
7. Create a Choices section with a bulleted list.
8. Create an empty list item for each bitmap image you want to display.
9. Import the **Correct Choice** fragment to indicate the correct answer.
If you have more list items than available images, the images will cycle back to the beginning and repeat.

★ Here's an example

The template appears first, followed by the quiz as seen by the student:

Quiz

Setup

Choices: Images, No Break

Images: swplogo.bmp, text.bmp, frac.bmp, math.bmp

Question

Statement

Click the icon which appears when the cursor is inside mathematics in *Scientific Notebook*.

Choices



correctchoice

1 Click the icon which appears when the cursor is inside mathematics in *Scientific Notebook*.



You can find this image select question in images.tex in the Quizzes directory of your program installation. Click here to try the quiz.

Creating Short-Answer Questions

With the latest version of the Exam Builder, you can include and automatically grade certain types of short-answer questions. The key is to create a Choices section that names a function for evaluating the student's response against a correct answer. The Choices section should contain two lines something like this:

```
InputField(MATH)
```

```
GradeProc: givecredit(response =  $x \ln x - x$ )
```

The first line creates an input field in the generated quiz in which the student can enter a response. The entry *MATH* means that the input is assumed to be mathematics. The second line has two parts:

- The first part identifies which **GradeProc** to use in grading the quiz.

A GradeProc is a function defined externally for use by the computational engine and designed to determine whether to give credit for an answer. In this example, the GradeProc is the **givecredit** function, which gives the student credit for the answer if the following argument is true.

- The second part states the argument to be evaluated by the function.

In this example, the mathematics name `response` represents the student's response. If it is equal to the expression on the right ($x \ln x - x$), the argument is true and the student is given full credit for the problem.

Note: For Maple, the **givecredit** function is defined in two files: `gradeprocs.def` and `gradeprocs.m`.

★ **Here's an example**

You can see how short answer questions work in this example.

Question

Setup

```
b := rand(1,10)
```

Statement

Evaluate $\int \ln bx dx$

Choices

```
InputField(MATH)
GradeProc: givecredit(response = x ln bx - x)
```

The quiz in its entirety appears in `ebinputfield.tex`. [Click here to try it out.](#)

Creating Exam Parts

We sometimes need to write exams divided into several parts, not all of which are similar in design. With the **Part** section, we can create a series of exam parts, each with its own setup. This way, the questions in one part might permute, while those in another part might not. Let's see how to create an exam with several parts.

► To create an exam with several parts

1. Start a new exam template with an Exam header.
2. Create a Part section:
 - a. Press **Enter**.
 - b. Apply a Heading 1 tag to the new paragraph.
 - c. Type **Part**.
 - d. Press **Enter**.
3. Create a Setup section for this part of the exam.

In the Setup section, place all the options that govern the generation of instances from this part of the exam template.
4. For each question in the part, create a Question section.

Include a Setup section, if necessary; create question statements and substatements; enter the possible choices; and provide answers and solutions for the students.

5. Repeat steps 2–4 for each part in the exam.
6. Save the exam template as a .tex file.
7. Create the .qiz file.

★ **Here's an example**

See Milestone 14 and try the quiz here. Note that Milestone 14 is also a good example of question substatements.

Chapter 5

Using Additional Algorithmic Generation Features

You can build exams by building questions that use algorithms that take advantage of the power these Exam Builder features:

- **Definitions created with := and =**
- **Random number functions**
- **Conditions**
- **Formulas and Seed keyword paragraphs**

Using Definitions Created with := and =

The Exam Builder has two kinds of definitions. Both may occur only in Setup sections:

- **Immediate** definitions are created using the symbol :=. They are evaluated only once as the exam template is loaded into **SWP** or **SNB**. We recommend that in most cases you use immediate definitions.
- **Delayed** definitions are created using the symbol =. They are re-evaluated every time the defined variable is used in the generated exam. Delayed definitions can be used to achieve special effects on occasion but are not generally as useful as immediate definitions.

For example, if you set up a question using the immediate definition $a := \text{rand}(10)$, the Exam Builder uses the expression to define a as a random number. The definition occurs once, when an exam file is generated. Every time the variable a is evaluated by the Exam Builder, its value is the same. The value of the variable will change only if you generate a new exam, at which time the Exam Builder may generate a new random value for a .

On the other hand, if you set up a question using the delayed definition $a = \text{rand}(10)$, the Exam Builder defines a as the random number expression itself. This expression is re-evaluated every time the variable a occurs, either in the statement setup or in the choices. Therefore, it's possible for the variable to have different values at different points in the same question.

Delayed evaluations can be useful in certain circumstances. Suppose you want to have a variable a that ranges between -10 and 10 and a variable b that is the reciprocal of a . Of course you must have $a \neq 0$. The first instinct might be to write a Setup section that looks like this:

```
a := rand(-10, 10)
```

```
b := 1/a
```

```
Conditions: a ≠ 0
```

The problem is that all assignments are done first and then the conditions are checked. So suppose that the first assignment causes a to be set to 0. Then the second assignment will cause an error, and the condition will never be checked. Thus the question being generated will fail once in every twenty-one generations. To avoid this problem, you could use $b = 1/a$ rather than the := form shown. This will cause b to be defined as $1/a$ but the evaluation will not take place until later when b is used in the question statement. By that time, the condition will have ensured that a is not 0.

Note Because the Exam Builder creates definitions automatically, you must not provide explicit definitions for variables, as you ordinarily do when you create mathematics in **SWP** or **SNB**. You must not use the Compute New Definition command to create questions in Exam Builder source files.

Using Random Number Functions: rand and randmat

In **SWP** and **SNB**, the basic random number function is called **rand**. You can combine random number functions with definitions to gain a powerful algorithmic tool. The **rand** function has four forms. An

additional random number function produces random matrices.

- `rand()`

The form `rand()`—with no argument—produces a random integer from a large range.

Example: `rand()` might yield a number such as 427419669081, 321110693270, or 343633073697.

- `rand(n)`

The form `rand(n)`, where *n* as a single positive integer, produces a random integer between 0 and *n* – 1.

Example: `rand(10)` produces one of these numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

- `rand(n, m)`

The form `rand(n, m)`, where *n* < *m*, produces a random integer between *n* and *m*, inclusive.

Example: `rand(2, 7)` produces one of these numbers: 2, 3, 4, 5, 6, or 7.

- `rand({some set})`

The form `rand({some set})` produces an element selected at random from the specified set.

Example: `rand({2, 4, 6, 8})` produces one of these numbers: 2, 4, 6, or 8.

This form of the function can also be used with expressions. The function `rand({ $\sin x$, $\cos x$ })` randomly selects either $\sin x$ or $\cos x$.

- `randmat`

An additional random number function, called `randmat`, produces random matrices of various types.

The function has this form:

$$\text{randmat}(a, b, c, (d))$$

where

a is the number of matrix rows,

b is the number of matrix columns,

c is the matrix type, with

1 = arbitrary

2 = symmetric (must be square)

3 = antisymmetric (must be square)

4 = unimodular and

(*d*) is the selection range for the values in the matrix:

The range (*n*), with *n* as a single positive integer, produces matrix values between 0 and *n* – 1.

The range (*n*, *m*), where *n* < *m*, produces matrix values between *n* and *m*, inclusive.


The range ({some set}) produces matrix values selected at random from the specified set.

Example: The function `randmat(4, 4, 1, (1, 9))` produces a 4×4 arbitrary matrix whose values are between 1 and 9, inclusive. Here's an example:

$$\text{randmat}(4, 4, 1, (1, 9)) = \begin{matrix} 7 & 8 & 8 & 4 \\ 1 & 3 & 4 & 3 \\ 4 & 4 & 8 & 9 \\ 4 & 6 & 1 & 6 \end{matrix}$$

► To enter a random number function

1. Open your **SWP** or **SNB** document.
2. Place the insertion point where you want the function.

3. Start mathematics.
4. Type type **rand** to enter a random number function or **randmat** to enter a random matrix. The program automatically recognizes the function and displays it on your screen in gray letters: rand or randmat.
5. Click  .
6. Complete the function argument.

You can also enter the functions by choosing them from the popup list in the Fragments toolbar or using the **Import Fragment** command on the **File** menu. More information about using mathematical functions in **SWP** and **SNB** is available via the online Help.

Using Conditions

If you simply generate a random number for use in a definition, the results can at times be unpredictable, but you can prevent such results if you specify conditions that randomly defined variables must meet. For example, you might set a condition that prevents a variable from being defined as 0 or a condition that prevents one variable from being greater than another.

When you use algorithms to state not only a question but also its multiple-choice answers, you can use conditions to prevent the random generation of duplicate or undefined answers. For example, a question that involves two randomly generated variables a and b might have as possible multiple-choice answers both $\frac{a}{b}$ and $\frac{b}{a}$. If the randomly generated values for the variables are the same, these answers will be identical and will present a confusing choice to the student. You can avoid the confusion by setting the condition that $(a \neq b)$. You also want to avoid the illegal situations of $(a = 0)$ and $(b = 0)$.

Like the definitions to which they refer, conditions can affect an entire exam or just a single question or question variant. Also like definitions, each condition must be stated on its own line in a Setup section. Further, each condition must follow the definition to which it refers. You can use logical and (\wedge) and logical or (\vee) to combine several conditions in a single statement.

Note that the conditions must be relatively easy to satisfy. If after several hundred attempts the Exam Builder can't find a random assignment to the variables that satisfies the conditions, it returns undefined results.

► To enter a condition

1. In the exam template, start a new paragraph after the definition for which you want to state conditions.
2. Type the **keyword Condition**: and then, with the insertion point in mathematics, type the condition the variable must meet.

Chapter 2 and the Keyword Reference explain more about keywords.

★ Here are some examples

- This condition specifies that neither a nor b can equal 0. The Exam Builder will reject a randomly generated value of 0 for either variable.

Condition: $(ab \neq 0)$

- This simple example shows how you can combine definitions, random number functions, and conditions to generate suitable variable values for your questions and answers.

$a := \text{rand}(10)$

$b := \text{rand}(10)$

$c := a + b$

Condition: $(a \neq b)$

This condition specifies that a and b cannot be equal. The Exam Builder will reject a randomly

generated value for b if it is the same as the value already generated for a .

- Here is a condition that is very hard to satisfy:

```
a := rand(1000000)
```

```
b := rand(1000000)
```

```
Condition: (a = b)
```

- This condition will not be satisfied very often. The chances are small that two numbers selected at random from the specified range will be equal.


Using Formulas

Once you have defined the variables for a particular question, you can use formulas to state the question and the corresponding multiple-choice answers. A formula is a special type of mathematical expression that is evaluated before it is displayed in your file. For example, if you enter a formula consisting of the simple expression $3 + 5$, **SWP** or **SNB** displays 8 in your file. Similarly, if you enter $(a + b)(a + b)$ as a formula, the program displays $(a + b)^2$ in your file. If you have **Helper Lines** turned on in the **View** menu, formulas appear against a colored background, as shown here. If you have **Helper Lines** turned off, formulas are visually indistinguishable from ordinary mathematical expressions.

Note that if the expression in a formula includes variables determined by random number functions or by certain mathematical operations, the program may not have all the information it needs to display the fully evaluated formula until it generates an exam (.qiz) file. For example, if as noted above you enter $(a + b)(a + b)$ as a formula and then use random number functions to define a and b , the program displays $(a + b)^2$ in your source file. It can't evaluate the expression further until the random values are assigned to the variables, which doesn't occur until the exam (.qiz) file is generated. At that time, if $a = 2$ and $b = 3$, the formula appears as 25 in the generated exam.

Note You should ignore the formula that appears in the Exam Builder exam template. It may differ significantly from both the statement of the formula and the fully evaluated form of the formula. You can see the formula in its unevaluated form if you double-click it.

- To create a formula

1. If the Field toolbar is displayed, click  .
or
From the **Insert** menu, choose **Formula**.
2. In the **Formula** area of the dialog box, enter a mathematical expression.
3. In the **Operation** area, enter the computational operation you want to perform on the expression.
Click the arrow at the right of the box for a list of available operations.
4. Choose **OK**.
If **Helper Lines** are turned on, the formula appears against a colored background.

See also the online Help topic on formula objects.

Using the Seed Keyword Paragraph

The **Seed** keyword can be placed in the Main Setup section of an exam to set an explicit random number generator seed. This technique is useful when debugging an exam. By setting the seed, you will get the same random numbers every time you generate the exam. If you change the seed number, you change the generated exam. If you remove the Seed keyword paragraph, the Exam Builder sets the seed by sampling the system clock at the time the exam is taken. Hence, you get a different quiz when you take the exam at different times.

Building Questions with Algorithms

Definitions, conditions, random number functions, and formulas all work together in Exam Builder questions and answers, as shown in the examples that follow.

★ **Here's the first example**

Define the variable n as a randomly selected prime number:

$n := \text{rand}(\{5, 7, 11, 13\})$

Ask the student to identify the prime number in a list of answers, each of which has been defined using formulas:

- n
- $2n$
- $3n$
- $n - 1$

★ **Here's another example**

Define several variables with random functions and restrict the definitions with conditions:

$a := \{2, 3, 5, 7, 11, 23\}$

$b := \text{rand}(a)$

$c := \text{rand}(a)$

Conditions: $(b \neq c) \wedge b < c$

State the question using several simple formulas:

If $x + b = c$, then x is:

Use additional formulas to determine several possible answers:

- $c - b$
- $b - c$
- $c - b + 1$

★ **Here's a final example**

This example is somewhat more complicated. Define several sets of variables with random number functions, then define additional variables using the first set of definitions:

$a := \text{rand}(1, 100)$

$b := \text{rand}(1, 100)$

$p := \text{rand}(\{3, 5, 7, 11, 13, 17, 19\})$ [some primes]

Condition: $\text{gcd}(a, p) = 1$

Condition: $\text{gcd}(b, p) = 1$

$A := -(b/a) \bmod p$

$B := -(b/a + 1) \bmod p$

$C := -(b/a - 1) \bmod p$

Use formulas to state the question for the student:

Solve the congruence $b + Xa = 0 \bmod p$.

Use additional formulas to present the possible answers:

- A
- B
- C



Chapter 6

Delivering Course Materials

You can deliver an Exam Builder quiz to the students in several ways:

- You can have your students **download the quiz** to their own computers. If they have **SWP** or **SNB**, they can open the .qiz file and take the quiz whenever they want and as many times as they want.
- You can **administer the quiz from your LAN or Windows NT server**. The actual quiz the students see will be generated and graded on the server. All attempts at downloading and grading will be recorded in a database.
- You can **generate and print several exams** and deliver them in class as usual.

We don't downloading a quiz here, because the process differs for everyone.

This chapter introduces these keywords: ItemID, Record, and Compute.

Administering Exams from a LAN or Server

Administering exams from a LAN or Server involves These steps:

1. Modifying the database.
2. Modifying the exam and placing it on the LAN or Server
3. Adding quizzes to switcher.tex.
4. Modifying the Submit button.
5. Examining the results in the database.

Step 1. Modify the database

Each quiz that you run on the server needs an ID that is unique, at least among quizzes whose scores are being recorded into the same database. The database must contain the ID for each quiz, as well as the names and passwords of your students.

► To add information to the database

1. In Microsoft Access, open the Students table of the database in Microsoft Access.
2. Add the names and passwords of the students to the database.
3. Open the Quizzes table and add the ID for each quiz.
4. Save and close the database.

Step 2. Modifying the exam and placing it on the LAN or Server

Each quiz must contain a unique ID to distinguish it from other quizzes whose scores are recorded in the database. Also, the quiz should contain instructions to the Exam Builder about whether to record the students' scores. Finally, the quiz should include instructions about whether to disable computational capability in **SWP** and **SNB** so that it can't be used to solve problems or leave it available during the exam. Use keyword paragraphs in the Main Setup section of the exam to give these instructions to the Exam Builder.

► To modify the exam

1. In **SWP** or **SNB**, open the .tex file for the exam.
2. Place the insertion point in the Main Setup section of the quiz
3. Add the ItemID keyword paragraph to the section:
 - a. If the insertion point isn't at the beginning of a paragraph, press **Enter** to start a new paragraph.
 - b. Enter the keyword **ItemID**:
 - c. Enter an ID that identifies the quiz uniquely within the database.

The example quizzes use these ItemIDs: FRACTIONS, EQUATIONS, and CENTROIDS.

- d. Press **Enter**.
4. Add the Record keyword paragraph to the section:
 - a. If the insertion point isn't at the beginning of a paragraph, press **Enter** to start a new paragraph.
 - b. Enter the keyword **Record**:
 - c. Enter a keyword:
 - Enter **Always** to record the score in the database each time the exam is taken.
 - Enter **Never** to prevent recording the score.
 - Enter **Student Choice** to allow the student to determine whether or not to record the score
 - d. Press **Enter**.
5. Add the Compute keyword paragraph to the section:
 - a. If the insertion point isn't at the beginning of a paragraph, press **Enter** to start a new paragraph.
 - b. Enter the keyword **Compute**:
 - c. Enter a keyword:
 - Enter **Disable** to turn off computational capability in **SWP** or **SNB**.
 - Enter **Enable** to turn on computational capability.
 - d. Press **Enter**.
6. Save the .tex file.
7. Save the .qiz file.
8. Move the .qiz file to the LAN or Server.

★ **Here's an example**

Click here to see an example of how these keyword paragraphs are used to set up an Exam Builder quiz in the **EBWeb\samples\Quizzes** directory.

Previous versions of the Exam Builder included the keywords HTTP, Grader, GradeMode, Handler, and RemoteHandler. These should no longer be used unless, for example, you want to write your own grader program.

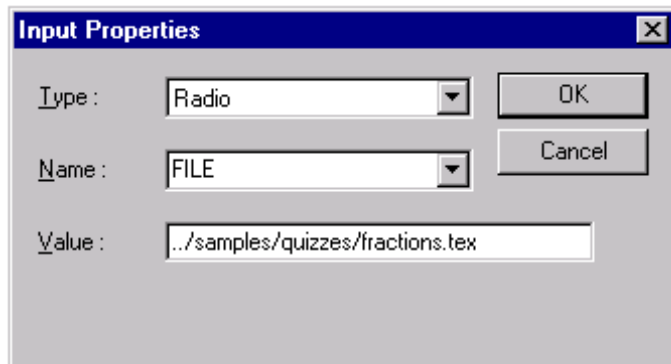
Step 3. Adding quizzes to switcher.tex

You must add each quiz you want to administer to the switcher.tex file in your **EBWeb\Samples** directory on the LAN or Server..

► To add quizzes to switcher.tex

1. Open switcher.tex and change it to a read/write file.
2. Copy and paste the radio buttons in the file as necessary to create new quiz items.
3. Double-click each radio button to change its properties.

When you double-click a radio button, the program opens the **Input Properties** dialog box, which indicates the kind of button and the name and path of the file in question.



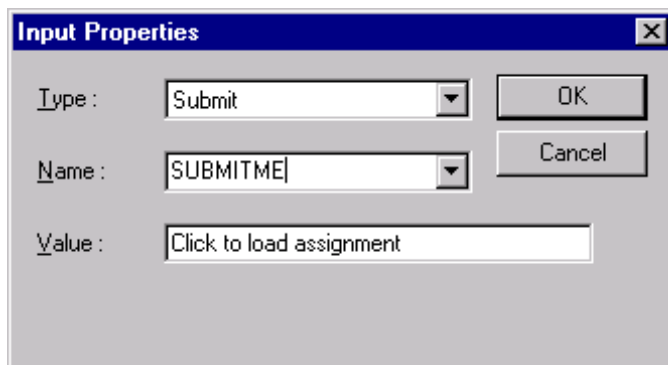
4. Change the Value to reflect the address of the indicated .qiz file.
The path is relative to your ebweb\bin directory: that is, it is relative to the location of the exam builder executable.
5. Choose **OK**.
6. Repeat steps 3–5 for each .qiz file you want to add.
7. Save switcher.tex as a read-only file.

Step 4. Modifying the Submit button

By default, the Submit button in switcher.tex appears as a gray box and carries the text *Click to load assignment*. You can change the text.

► To change the text on the Submit button

1. Open switcher.tex and change it to a read/write file.
2. Double-click the Submit button to change its properties.



3. Edit the values as necessary.
4. Choose **OK**.
5. Save switcher.tex as a read-only file and close the file.

Step 5. Examining the results in the database

As the students take the quiz, the Exam Builder grades their work and records the scores in the database

► To examine scores in the database

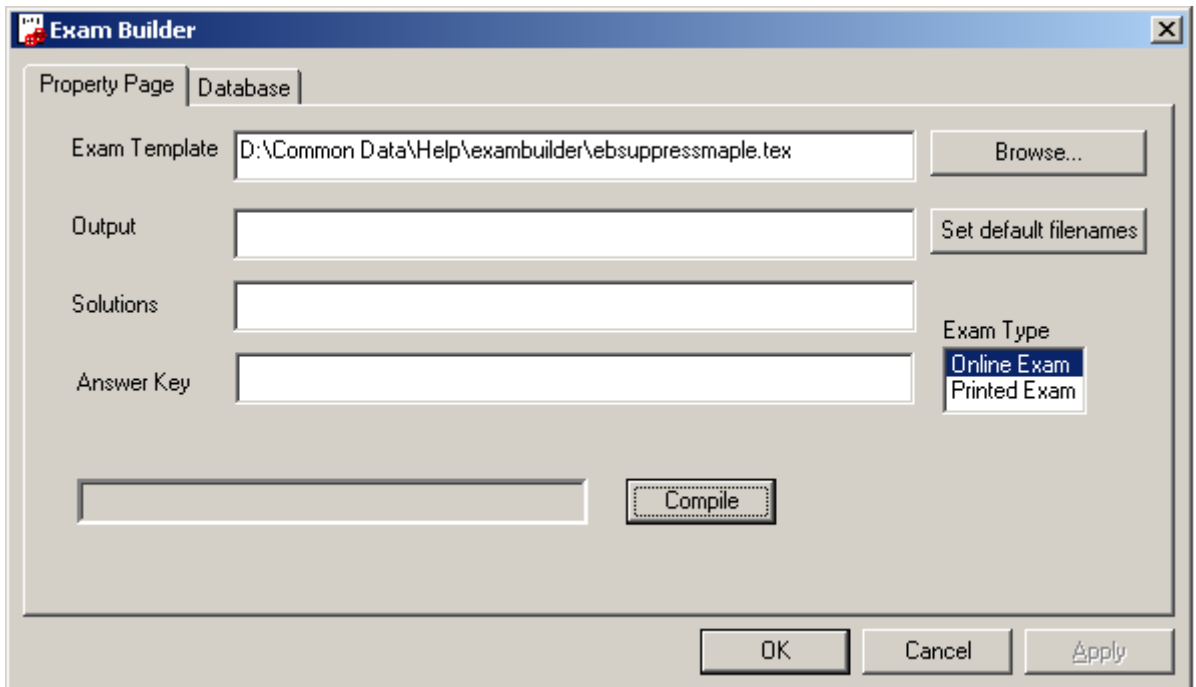
1. In Microsoft Access, open the ...\\EBWeb\\Samples\\databases\\samples.mdb database.
If you have Access 2000, the program may ask whether you want to convert the database to the format expected by the Exam Builder. Choose **No**.
2. Open the Activity table to find the scores.

You can use the Convert button on the Database tab of the Exam Builder dialog box to convert a database to a file of the same name but with a comma-separated value (.csv) format, which is a plain text format that can be read by Microsoft Excel and other applications. You can use Excel to view your scores but not to edit the database.

Generating Printed Exams

If delivering exams online isn't an option, you can generate and print copies of your completed exams for distribution to your students. If you want to create a lot of printed quizzes, you might want to write a batch file to do the job, as described in the Reference. Follow these steps to print a generated exam:

- ▶ To generate printed versions of an exam
 1. Develop and test the exam template.
 2. In the Main Setup Section, add a keyword paragraph with a **Print Choices** command. The command tells the Exam Builder to substitute the items you list in the Print Choices paragraph for any radio buttons and check boxes in the template.
 - a. Start a new paragraph.
 - b. Type the keyword **Print Choices**:
 - c. Type **a, b, c, d, e, f**
 3. Save the template as a .tex file.
 4. Create the .qiz file.
 5. Start the Exam Builder as a separate application:
 - Run **ExamBld2.exe** from a DOS prompt.
 - or*
 - If you've created a desktop icon for the Exam Builder, click the icon. The Exam Builder opens this dialog box:



6. Complete the entries in the dialog box:
 - a. In the Exam Template box, type the path name of your exam template.

- b. Press **Set default filenames** to have the Exam Builder generate automatic names for the Output, Solutions, and Answers files.
- c. If you don't want default file names,
 - i. In the Output box, type a new file name for the generated exam. Because you will open this file in **SWP** or **SNB**, create the file with a **.tex** extension.
 - ii. In the Solutions box, type the name of the solutions file.
 - iii. In the Answer Key box, type the name of the answer key file.
- d. In the Exam type area, choose **Printed Exam**.
- e. Choose Compile.

The Exam Builder generates the exam, the answer key, and the solutions file, if any. The answer key is saved in a file named the same as the file specified in the Output Filename area but with the word *key* added. For instance, if the output file is *out.tex*, the answer key is named *out key.tex*.

7. In **SWP** or **SNB**, open and preview the exam file, the answer key, and the solutions file.
8. Print the files.



Chapter 7

Reference Materials

These materials summarize and expand on Exam Builder information introduced in earlier chapters.

- Keywords

 - Keyword Headings

 - Keyword Paragraphs

 - Keyword Lists

- Algorithmic Generation Reference

- Exam Builder Command Line Interface

- Troubleshooting



Reference

Keywords

Keyword Headings

When the Exam Builder encounters a keyword in a section heading, it associates all subsequent material with that keyword until it encounters another keyword in another section heading. Otherwise, the Exam Builder ignores headings and heading levels entirely. In particular, the Exam Builder does not use a section heading containing a keyword to create a corresponding heading in the generated exam. The Exam Builder recognizes these keywords when they occur in headings:

Keyword	Effect
Answer	Indicates the correct answer to a question; displayed after grading
Assignment	Identifies a file as an Exam Builder template file
Choices	Begins the list of possible question answers
Comment	Identifies information that appears only in the template file
Exam	Identifies a file as an Exam Builder template file
Homework	Identifies a file as an Exam Builder template file
Part	Begins a part of a multipart exam
Question	Begins a question
Quiz	Identifies a file as an Exam Builder template file
Response	Identifies the area in which students enter a free-form response to a question
Setup	Begins a section in which certain values and variables are set
Solution	Begins a detailed explanation of a problem solution; displayed after grading
Statement	Presents the text of a question
Substatement	Forms multipart questions
Test	Identifies a file as an Exam Builder template file
Text	Begins a section of literal text to be included in the generated exam
Tutorial	Identifies a file as an Exam Builder template file
Variant	Begins an alternative to the question in which the variant occurs

Keyword Paragraphs

In a Setup section, the Exam Builder recognizes certain keywords. A keyword must occur at the start of a paragraph and be followed by a colon and then by a list of additional keywords that serve as parameters. The Exam Builder associates each keyword in the list with the keyword at the beginning of the paragraph.

The Exam Builder recognizes these keywords when they occur at the beginning of a paragraph:

Keyword	Effect
Action	Indicates a program or script you want to run on a remote server
Choice Space	Establishes the horizontal space between multiple-choice answers
Choices	Begins a set of specifications for multiple-choice answers
Compute	Turns computation engine on or off during exam taking (enable, disable)
Condition	States a condition the random variables in a setup section must satisfy
CSTFile	Specifies a style file for formatting the exam
Database	Names the Microsoft Access database used to record scores
GradeProc	Names a procedure to use for grading a short answer question
Inputfield	Creates an input field for a short answer question
Images	Names .bmp or .dib graphics to be inserted instead of radio buttons
ItemId	Indicates the ID used to identify this quiz in the database
Method	Names a CGI method, usually set to POST
Points	Assigns a point value to an exam question
Print Choices	Specifies the elements that appear instead of radio buttons when multiple-choice questions are printed
Question Space	Establishes the vertical space between questions
Questions	Begins a set of specifications for questions
Record	Does or doesn't record scores (Never, Always, Student Choice)
Seed	Sets a seed value for random number generation
Select	Specifies the number of question variants to include on an exam
Submit	Indicates a label for the Submit button
Text Area	Defines the size of a free-form response area
Title	Contains information to be displayed on the title bar when a generated exam is viewed online

Keyword Lists

Certain keyword paragraphs can be associated with one or more keywords that follow the colon at the end of the keyword. The Exam Builder associates the keywords in the list with the keyword at the beginning of the paragraph. When several of these keywords occur in a list, they must be separated by commas.

Keyword	Keyword	Effect
Paragraph		
Choices	Break	Places multiple-choice answers on separate lines
Choices	Check	Places a checkbox next to each choice in a multiple-choice list
	No	Prevents effects when used in combination with other keywords such as Break , Permute or Number
Choices	Permute	Changes the order in which bulleted multiple-choice items appear in a list of choices or the order in which questions appear on the exam
Choices	Radio Buttons	Places a clickable button next to each choice in a multiple-choice list
Compute	Enable	Enable the computational engine in SWP or SNB while a quiz is being taken
Compute	Disable	Disable the computational engine in SWP or SNB while a quiz is being taken
Questions	Number	Numbers the questions on an exam
Questions	Permute	Changes the order in which bulleted multiple-choice items appear in a list of choices or the order in which questions appear on the exam



Reference

Algorithmic Generation Reference

Definitions := and =

Immediate definitions are created using the symbol :=. They are evaluated only once as the exam template is loaded into **Scientific WorkPlace** or **Scientific Notebook**. We recommend that in most cases you use immediate definitions.

Delayed definitions are created using the symbol =. They are re-evaluated every time the defined variable is used in the generated exam. Delayed definitions can be used to achieve special effects on occasion but are not generally as useful as immediate definitions.

For example, if you set up a question using the immediate definition $a := \text{rand}(10)$, the Exam Builder uses the expression to define a as a random number. The definition occurs once, when an exam file is generated. Every time the variable a is evaluated by the Exam Builder, its value is the same. The value of the variable will change only if you generate a new exam, at which time the Exam Builder may generate a new random value for a .

On the other hand, if you set up a question using the delayed definition $a = \text{rand}(10)$, the Exam Builder defines a as the random number expression itself. This expression is re-evaluated every time the variable a occurs, either in the statement setup or in the choices. Therefore, it's possible for the variable to have different values at different points in the same question!

Both kinds of definitions may occur only in Setup sections.

Because the Exam Builder creates definitions automatically, you must not provide explicit definitions for variables, as you ordinarily do when you create mathematics in **Scientific WorkPlace** or **Scientific Notebook**. You must not use the Compute New Definition command to create questions in Exam Builder source files.

As an example of where you want to use the delayed evaluation suppose you want to have a variable a that ranges between -10 and 10 and a variable b that is the reciprocal of a . Of course you must have $a \neq 0$. The first instinct might be to write a Setup section that looks like this:

$a := \text{rand}(-10, 10)$

$b := 1/a$

Conditions: $a \neq 0$

The problem is that all assignments are done first and then the conditions are checked. So suppose that the first assignment causes a to be set to 0. Then the second assignment will cause an error, and the condition will never be checked. Thus the question being generated will fail once in every twenty-one generations. To avoid this problem you could use $b = 1/a$ rather than the := form shown. This will cause b to be defined as $1/a$ but the evaluation will not take place until later when b is used in the question statement and by that time the condition will have ensured that a is not 0.

Random number functions: rand and randmat

In **Scientific WorkPlace** and **Scientific Notebook**, the basic random number function is called **rand**. You can combine random number functions with definitions to gain a powerful algorithmic tool. The rand function has four forms. In addition, an additional random number function produces random matrices.

- `rand()`

The form `rand()`—with no argument—produces a random integer from a large range.

Example: `rand()` might yield a number such as 427419669081, 321110693270, or 343633073697.

- `rand(n)`

The form $\text{rand}(n)$, where n as a single positive integer, produces a random integer between 0 and $n - 1$.

Example: $\text{rand}(10)$ produces one of these numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

- $\text{rand}(n, m)$

The form $\text{rand}(n, m)$, where $n < m$, produces a random integer between n and m , inclusive.

Example: $\text{rand}(2, 7)$ produces one of these numbers: 2, 3, 4, 5, 6, or 7.

- $\text{rand}(\{\text{some set}\})$

The form $\text{rand}(\{\text{some set}\})$ produces an element selected at random from the specified set.

Example: $\text{rand}(\{2, 4, 6, 8\})$ produces one of these numbers: 2, 4, 6, or 8.

This form of the function can also be used with expressions. The function $\text{rand}(\{\sin x, \cos x\})$ randomly selects either $\sin x$ or $\cos x$.

- randmat

An additional random number function, called randmat , produces random matrices of various types. The function has this form:

$$\text{randmat}(a, b, c, (d))$$

- where

- a is the number of matrix rows,
- b is the number of matrix columns,
- c is the matrix type, with
 - 1 = arbitrary
 - 2 = symmetric (must be square)
 - 3 = antisymmetric (must be square)
 - 4 = unimodular and

(d) is the selection range for the values in the matrix:

The range (n) , with n as a single positive integer, produces matrix values between 0 and $n - 1$.

The range (n, m) , where $n < m$, produces matrix values between n and m , inclusive.

The range $(\{\text{some set}\})$ produces matrix values selected at random from the specified set.

Example: The function $\text{randmat}(4, 4, 1, (1, 9))$ produces a 4×4 arbitrary matrix whose values are between 1 and 9, inclusive. Here's an example:

$$\text{randmat}(4, 4, 1, (1, 9)) = \begin{matrix} 7 & 8 & 8 & 4 \\ 1 & 3 & 4 & 3 \\ 4 & 4 & 8 & 9 \\ 4 & 6 & 1 & 6 \end{matrix}$$

► To enter a random number function

1. Open your **Scientific WorkPlace** or **Scientific Notebook** document.
2. Place the insertion point where you want the function.
3. Start mathematics.
4. Type type **rand** to enter a random number function or **randmat** to enter a random matrix. The program automatically recognizes the function and displays it on your screen in gray letters: rand or randmat .

5. Click  .
6. Complete the function argument.

You can also enter the functions by choosing them from the popup list in the Fragments toolbar or using the **Import Fragment** command on the **File** menu. You can find more information about using mathematical functions in *Scientific WorkPlace* and *Scientific Notebook* is available online via the Help Contents.

Using conditions

If you simply generate a random number for use in a definition, the results can at times be unpredictable, but you can prevent such results if you specify conditions that randomly defined variables must meet. For example, you might set a condition that prevents a variable from being defined as 0 or a condition that prevents one variable from being greater than another.

When you use algorithms to state not only a question but also its multiple-choice answers, you can use conditions to prevent the random generation of duplicate or undefined answers. For example, a question that involves two randomly generated variables a and b might have as possible multiple-choice answers both $\frac{a}{b}$ and $\frac{b}{a}$. If the randomly generated values for the variables are the same, these answers will be identical and will present a confusing choice to the student. You can avoid the confusion by setting the condition that $(a \neq b)$. You also want to avoid the illegal situations of $(a = 0)$ and $(b = 0)$.

Like the definitions to which they refer, conditions can affect an entire exam or just a single question or question variant. Also like definitions, each condition must be stated on its own line in a Setup section. Further, each condition must follow the definition to which it refers. You can use logical and (\wedge) and logical or (\vee) to combine several conditions in a single statement.

Note that the conditions must be relatively easy to satisfy. If after several hundred attempts the Exam Builder can't find a random assignment to the variables that satisfies the conditions, it returns undefined results.

► To enter a condition

1. In the exam template, start a new paragraph after the definition for which you want to state conditions.
2. Type the **keyword Condition**: and then, with the insertion point in mathematics, type the condition the variable must meet.
Chapter 2 and the Keyword Reference explain more about keywords.

★ Here are some examples

- This condition specifies that neither a nor b can equal 0. The Exam Builder will reject a randomly generated value of 0 for either variable.

Condition: $(ab \neq 0)$

- This simple example shows how you can combine definitions, random number functions, and conditions to generate suitable variable values for your questions and answers.

$a := \text{rand}(10)$

$b := \text{rand}(10)$

$c := a + b$

Condition: $(a \neq b)$

This condition specifies that a and b cannot be equal. The Exam Builder will reject a randomly generated value for b if it is the same as the value already generated for a .

- Here is a condition that is very hard to satisfy:

$a := \text{rand}(1000000)$

$b := \text{rand}(1000000)$

Condition: $(a = b)$


- This condition will not be satisfied very often because what are the chances that two numbers selected at random from the specified range will be equal?

Using formulas

Once you have defined the variables for a particular question, you can use formulas to state the question and the corresponding multiple-choice answers. A formula is a special type of mathematical expression that is evaluated before it is displayed in your file. For example, if you enter a formula consisting of the simple expression $3 + 5$, **Scientific WorkPlace** or **Scientific Notebook** displays 8 in your file. Similarly, if you enter $(a + b)(a + b)$ as a formula, the program displays $(a + b)^2$ in your file. If you have **Helper Lines** turned on in the **View** menu, formulas appear against a colored background, as shown here. If you have **Helper Lines** turned off, formulas are visually indistinguishable from ordinary mathematical expressions.

Note that if the expression in a formula includes variables determined by random number functions or by certain mathematical operations, the program may not have all the information it needs to display the fully evaluated formula until it generates an exam file. For example, if as noted above you enter $(a + b)(a + b)$ as a formula and then use random number functions to define a and b , the program displays $(a + b)^2$ in your source file. It can't evaluate the expression further until the random values are assigned to the variables, which doesn't occur until the exam file is generated. At that time, if $a = 2$ and $b = 3$, the formula appears as 25 in the generated exam. You should ignore the formula that appears in the Exam Builder exam template, which may differ significantly from both the statement of the formula and the fully evaluated form of the formula. You can see the formula in its unevaluated form if you double-click it.

► To create a formula

1. If the Field toolbar is displayed, click  .
or
From the **Insert** menu, choose **Formula**.
2. In the **Formula** area of the dialog box, enter a mathematical expression.
3. In the **Operation** area, enter the computational operation you want to perform on the expression.
Click the arrow at the right of the box for a list of available operations.
4. Choose **OK**.
If **Helper Lines** are turned on, the formula appears against a colored background.

See also the main Help topic on formula objects.

The Seed Keyword Paragraph

The **Seed** keyword can be placed in the Main Setup Section of an exam to explicitly set the random number generator seed. This is useful when debugging an exam. By setting the seed, you will get the same random numbers every time you generate the exam. If you change the seed number, you change the generated exam. If you remove the Seed keyword paragraph, then the seed is set by sampling the system clock at the time the exam is taken. Hence, you get a different quiz when you take the exam at different times.

Building Questions with Algorithms

Definitions, conditions, random number functions, and formulas all work together in Exam Builder questions and answers, as shown in the examples that follow.

★ **Here's the first example**

Define the variable n as a randomly selected prime number:

$n := \text{rand}(\{5, 7, 11, 13\})$

Ask the student to identify the prime number in a list of answers, each of which has been defined using formulas:

- n
- $2n$
- $3n$
- $n - 1$

★ **Here's another example**

Define several variables with random functions and restrict the definitions with conditions:

$a := \{2, 3, 5, 7, 11, 23\}$

$b := \text{rand}(a)$

$c := \text{rand}(a)$

Conditions: $(b \neq c) \wedge b < c$

State the question using several simple formulas:

If $x + b = c$, then x is:

Use additional formulas to determine several possible answers:

- $c - b$
- $b - c$
- $c - b + 1$

★ **Here's a final example**

This example is somewhat more complicated. Define several sets of variables with random number functions, then define additional variables using the first set of definitions:

$a := \text{rand}(1, 100)$

$b := \text{rand}(1, 100)$

$p := \text{rand}(\{3, 5, 7, 11, 13, 17, 19\})$ [some primes]

Condition: $\text{gcd}(a, p) = 1$

Condition: $\text{gcd}(b, p) = 1$

$A := (-b/a) \bmod p$

$B := -(b/a + 1) \bmod p$

$C := -(b/a - 1) \bmod p$

Use formulas to state the question for the student:

Solve the congruence $b + Xa = 0 \bmod p$.

Use additional formulas to present the possible answers:

- A
- B
- C



Reference

Exam Builder Command Line Interface

Running from a Command Prompt

The Exam Builder executable is called **eb.exe**. It is located in the main program directory. By placing this directory in your path setting, you can run the Exam Builder from a command prompt. Here is the syntax:

```
EB inputfilename [/D] [/OUT:outputfilename] [/SLN:solutionfilename] [/KEY:answerfilename]  
>redirectedoutput
```

/D brings up the dialog box interface.

/OUT specifies the output (.qiz) file.

/SLN specifies the solutions file.

/KEY specifies the answer key file.

Creating a Batch File

If you have to create a lot of printed quizzes, you can write a batch file to do it using the command line interface. By placing the main program directory in your path setting, you can run the Exam Builder from a command prompt. Here is the command line syntax:

```
EB [/D] [/OUT:outputfilename] [/SLN:solutionfilename] [/KEY:answerfilename] inputfilename  
>redirectedoutput
```

/D brings up the dialog box interface

/OUT specifies the output (.qiz) file.

/SLN specifies the solutions file.

/KEY specifies the answer key file.



Reference

Troubleshooting the Exam Builder

Generating Error Messages

The Exam Builder can detect certain syntactic errors when compiling a template. If the Exam Builder detects an error in the template, it will not generate an exam instance. Instead, it will return a copy of the template marked up with bright red error messages. Click [here](#) to see what happens when you try to load a .qiz with syntax errors; here is the template.

You must correct the error before the program can create an exam instance. When you use a quiz created with Version 3.0, you are likely to encounter errors of the sort demonstrated in that template. because this version of the Exam Builder is less tolerant of errors than earlier versions. So when the new version indicates an error, you should fix the offending part of the template.

Finding Errors

Spaces in keyword headings and other constructs can cause problems. For example, the following heading has a vertical space (a small, green vertical arrow) just before the letter T; to see it, turn on **Invisibles** from the **View** menu.

This is a heading with a vertical space at the beginning

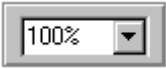
Another spacing error appears in the following mathematics, which looks correct. However, if you turn on invisibles, you see that a space between the a and the left parenthesis: $x + y = a (u - v)$ You may have to increase screen magnification to see the space.

Errors can also occur if you create algorithmically generated questions with conditions that might not be satisfied.

You can detect these errors while you're developing your quiz if you modify the view settings and modify the Error Handling defaults

- ▶ To detect spacing errors more easily

- Click  or, from the **View** menu, turn on Invisibles.

- Click  or, from the **View** menu, choose **Custom** and set the working view to 150% or more.

- ▶ To detect unsatisfied conditions

1. From the **Tools** menu, choose **Engine Setup**.
2. Choose the **Error Handling** tab.
3. Set _____
4. Choose **OK**.

This modification displays a message if a question is generated without the associated conditions being satisfied.

Working on a Server

To run the Exam Builder successfully on your server, you may need significant knowledge of web server operation along with assistance from your system administrator.

Be sure that CTI scripting is turned on for the Server. Also, make sure your Server has the correct permissions to run the program. Check the :iusr permissions.