ExamView Dynamic Questions Training Guide for PC and Mac Users

eInstruction

Simple Solutions. Real Results.

Introduction to Building Dynamic Content
ExamView Level 1 Training Guide

Contact Information

In addition to this training guide and the ExamView software help menu, following are other resources available to support your implementation of ExamView.

Sales Support: 800.530.0527
Technical Support: 800.530.0527
Technical Support Email: examview.support@einstruction.com
Downloadable Files/Videos: www.fscreations.com/download_center.php
User Community: forum.fscreations.com
Module 1: Introduction to ExamView and Dynamic Content

Overview

Participants will learn the basic attributes of ExamView and dynamic content.

Objectives

After completing this module, participants will be able to:

- Open the ExamView Test Generator
- Define a variable
- Format a variable’s attributes
- Insert a variable into an existing question
- Create a constant
What is the ExamView Assessment Suite?

The ExamView Assessment Suite is a family of software products that creates assessment questions and tests for K-12 classrooms, universities, and corporations.

- The ExamView Assessment Suite is comprised of the Test Generator, Test Manager, and Test Player:
  - **Test Generator**: Create and manage tests, quizzes, and worksheets using content from question banks.
  - **Test Manager**: View reports of test results and manage a class gradebook.
  - **Test Player**: Take tests that have been placed on a local area network (LAN).

- The ExamView Assessment Suite can save you hours of work each week and significantly improve how you create and administer tests, worksheets, and study guides, resulting in improved student learning and more accurate student progress assessment.

What is the ExamView Test Generator

The ExamView Test Generator is a powerful assessment tool for creating questions and building tests, worksheets, and study guides in any subject area. The ExamView Test Generator includes two components: Question Bank Editor and Test Builder.

- The Question Bank Editor allows you to create your own question banks or to edit questions in the ExamView Learning Series (22 grade-level products with more than 8,400 questions in math, science, social studies, and reading) or an existing publisher-supplied ExamView formatted question bank.
  - A question bank (or a bank) is a database of up to 250 questions from which you can make selections to be included on a test.
  - A question bank can include questions from any of the 14 different question types, including multiple choice, true/false, short answer, and completion.

- The Test Builder allows you to create a test from questions in the Question Bank Editor and to print and save a test.
  - A test is a series of formatted questions of one or more question types.
- Question banks are the source files of questions that appear on a test.
  - You should make any permanent changes to questions at the source (i.e., in the question bank), not on a test.
  - If you make changes to questions on a test, those changes will not be reflected in the question bank.
Introduction to Building Dynamic Content

Start ExamView Test Generator

1. Double-click the **ExamView Test Generator** shortcut icon on the desktop. The ExamView Test Generator – Startup dialog opens.

   *Note.* If this is the first time you are running ExamView Test Generator, you will be prompted to enter your name, organization and city/state.

2. Select **Open an existing question bank**.

3. Navigate to the **Training folder**.

4. Select **Dynamic Bank.bnk**. Then, click **Open**.

Introduction to Dynamic Content

The ExamView Test Generator allows you to create powerful dynamic content.

- Dynamic content contains algorithms that use programmed logic and commands to change variables within the question stem, answers, and, in many cases, the rationale.
- Dynamic content utilizes variable ranges rather than static values to automatically produce multiple questions that test the same concept.
- A question is dynamic if it has a dynamic tag ( ) to the left of the question number.
- Algorithms allow infinite customization of a test question using mathematical definitions.
  - For example, with a single click, you can print multiple versions of the same test, with the ExamView Test Generator automatically calculating new values for every question and answer.
- Through the use of mathematical, string, and special functions along with the ability to create custom functions, you can create dynamic content for any subject area.

Change Values of Dynamic Questions

1. Click the **View** menu. Confirm that **View > Question Tags** is selected.

   *Note.* When View | Question Tags is selected, all questions with dynamic content will have a dynamic tag ( ) to the left of the question number.

2. Select the first question and click **calculate values**. The Calculate Values feature causes the dynamic content to change or to recalculate.
3. Select the **Multiple Choice section** header and click **calculate values** to change the values of all the dynamic questions in the multiple choice section.

4. Select the **Test Title** header and click **calculate values** to change the values of all the dynamic questions in the entire test.

**Explore a Dynamic Question**

1. Double-click **Multiple Choice 1**. The Multiple Choice question window opens. Note the shaded words throughout the question stem, the answers, and the rationale. A shaded word (called a variable) is dynamic content.

2. Click **Calculate Values** twice. Note the changes in the question and answer choices.

3. Click **Close**.

4. Click **No** when asked to record the question.

5. Choose **File > Close** to close the question bank.

6. Choose **No** to close the question bank without saving it.

**Define a Variable**
Dynamic content is built on variables.

- A variable is a named object that can be modified during recalculation.
- Each variable has a unique name and a “scope” or context where it is identified.
  - In the ExamView Test Generator, the scope of a variable is either limited to a single question, a narrative and all questions linked to it, or to a matching group and all questions within that matching group.
- To define a variable, remember the following guidelines:
  - Each variable name must be unique for the algorithm group and must not match any of the keywords.
  - Certain variable names are reserved and should not be used in certain situations. For example, you cannot use the names of the built-in functions (e.g., cos) as variable names.
  - A variable name must begin with an alphabetic character.
  - A variable name cannot be longer than 40 characters.
  - A variable name cannot contain characters other than alphabetic characters, numeric characters, and the underscore.
  - Use variable names that represent the variable. This will make it easier to review your list of variables and know immediately what each variable represents in the question.
- There are three types of variables:
  - An integer variable includes positive and negative whole numbers and zero.
  - A double variable includes a double-precision floating point (i.e., the variable has a fractional part).
  - A string variable has characters or letters.
- You don’t have to explicitly assign a type to any variable or a constant. The program does that for you.

**Instructional Application**
In your own words, explain the concept of a variable.

_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________
_____________________________________________________________________________

Define a variable
1. Open **Static bank.bnk** from the **Training** folder.

2. Double-click Multiple Choice 1. The Multiple Choice question window opens.

3. Choose **Edit > Algorithm Definitions**. The Edit Algorithm Definitions window opens.

4. Click **New**. The New Algorithm window opens.

5. Click ː next to the variable definition to display a list of currently defined variables, constants, and keywords. You will use the algorithmic syntax rules to help you create the variable description. (See Syntax in the appendix of this manual.)

6. Select **Variable** from the Type drop-down list.

7. Enter **number1** in the Variable name field.

8. In the Definition field, enter **range(2,5,.05)** for the variable definition. (See Figure 3.)

9. Leave the New Algorithm window open for use in the next section, “Format the Variable.”

**Range Function**

- The function range chooses a random number from the first parameter to the second parameter with increments of the third parameter.
- If the increment is in the form .1, .01, .001, etc., then the result will always have the same number of decimal places as the increment.
- If the first parameter is less than zero, the second parameter is greater than zero, and an increment is provided, then the value 0 will never be returned by the function.
- Assume you want random numbers from 2 to 5 with increments of .05.

**Format a Variable**

You must choose how the variable will be formatted when displayed.
• If you choose a format type of number or scientific notation, you must specify the number of
decimal places you want displayed and whether you want this to be a fixed number.
• You can choose between General, Number, Comma Separated Number, Scientific Notation,
Coefficient, Coefficient (1st), or Metric. (Later in this session, you will learn about the coefficient
and coefficient (1st) options.)
  o The General formatting option may be used for strings or numbers. You may not specify a
    number of decimal places or whether the decimal should be fixed or float.
  o The Number, Number(#,###,###), Metric (# # # #), and Scientific Notation formatting
    options allow you to specify the number of decimal places, whether the decimal place
    should be fixed or float, and whether to show or hide leading zeroes.
    ▪ Specifying a smaller number of decimal places than the number contains will cause
      the number to be rounded at the time the number is defined. This rounded number
      will be used in further calculations.
    ▪ The number 23456.126 formatted as Number with 2 decimal places floating or fixed
      will result in 23456.13.
  o The Number(#,###,###) format will provide commas in the number as well.
    ▪ The number 23456.126 formatted as Number (#,###,###) with 4 decimal
      places that are floating will result in 23,456.126.
    ▪ The number 23456.126 formatted as Number (#,###,###) with 4 decimal
      places that are fixed will result in 23,456.1260.
  o The Metric (# # # #) formatting option is similar in every respect to
    Number(#,###,###) except it uses spaces in place of commas. The one exception is that
    the number must be greater than or equal to 10,000 to include spaces.
    ▪ The number 23456.126 with 2 decimal places that are fixed would display as 23
      456.13 in the Metric (# # # #) format.
  o The Scientific Notation formatting option treats fixed and floating decimal places similar to
    the Number, Number(#,###,###), and Metric (# # # #). The number generated will be
    in Scientific notation format.
    ▪ The number 23456.126 with 5 fixed decimal places in scientific notation format
      would be displayed as 2.34561e+004.
  o The Coefficient and Coefficient (1st) formatting options are designed for numbers that are
    attached to a variable by multiplication.
• In the equation 3x + 4y = 12, the numbers 3 and 4 would be considered coefficients to the
  variables x and y. The number 3 would be formatted as Coefficient (1st) and the number 4
  would be formatted as Coefficient.
  o Both Coefficient and Coefficient (1st) formatting options will hide the number 1. 1x – 1y =
    6 would appear as x - y = 6.
  o The Coefficient formatting option will display the sign of the number. If the coefficient of y
    is -4 and it was formatted as Coefficient, it would display as x - 4y = 16.
• If the variable is a string variable, select General from the Format drop-down list.

Format a Variable

1. From the **Format** drop-down list, choose **Number**.

2. From the **Decimal places** drop-down list, **select 2**.

3. **Select Fixed** in order to display two decimal places representing the cents value.
Introduction to Building Dynamic Content

• Use the **Hide leading zero** option to display decimal numbers without a zero before the decimal point.
• When you duplicate a question, the ExamView Test Generator requires that at least one of the algorithms defined for the question is different for the new question.
• At times, you may want one or more variables to be unique, while other variables don’t matter.
• Use the **Make only this variable value unique across questions** option to mark variables which are critical to be unique across questions.

4. Click **OK**.
5. Click **Done**.

*Note.* If you need to make changes to a variable after it has been recorded, you can choose **Edit > Algorithm Definitions**, select the variable that you want to edit, and choose **Edit**.

**Insert a Variable**

You must insert the defined variables into the question, answer choices, and answers.

1. Highlight **5.00** in the question stem.
2. Choose **Insert > Variable**. The Insert Variable window opens.
3. Click **Insert**. The variable number1 is inserted into the question.
4. Choose View > Variable Names. You will see ?? appear in place of the variable name.

5. Click Calculate Values. A value for number1 replaced the ?? in the question.

Add Variable Comments and Save Question Bank

1. Double-click the current value for number1. The Format Variable window opens.

2. Press the RIGHT ARROW key to move to the end of the definition field.

3. Enter #original value and click OK.

4. Click Calculate Values to recalculate a value for number1.
5. Click **Record** to record the question.

6. Choose **File > Save As** to save and rename the question bank.

7. Enter **New Dynamic.bnk** in the File name field and click **Save**.

**Notes**

_____________________________________________________________________________

_____________________________________________________________________________

_____________________________________________________________________________

_____________________________________________________________________________

_____________________________________________________________________________

_____________________________________________________________________________

_____________________________________________________________________________

**Create a Constant**

- To define `number2`, `number3`, `number4`, and `number5`, you will add the same value to each preceding number to obtain the next number. That value will be a constant.
- A constant is a named object that retains a constant value through multiple recalculations.
- A constant can be a string or numeric literal, a combination of pre-defined constants, or an expression consisting of a combination of keywords and string or numeric constants.
- The rules for scope and naming conventions for a constant are the same as those for a variable.
• A constant cannot be inserted into a question, narrative, or matching group.
• There are eight constants permanently in the Test Generator:
  
  e    false    inf    no    pi    testversion    true    yes

1. Double-click **Multiple Choice 1** to open the edit window.

2. Choose **Edit > Algorithm Definitions**. The Edit Algorithm Definitions window opens.

3. Click **New**. The New Algorithm window opens.

4. From the **Type** drop-down list, select **Constant**.

5. Enter **increase** in the Constant name field.

6. Enter **.25** in the Definition field.

7. Click **OK**.
Module 2: Using Arithmetic Operators

Overview
Participants will learn the basic attributes of ExamView and dynamic content.

Objectives
After completing this module, participants will be able to:

- Open the ExamView Test Generator
- Define a variable
- Format a variable’s attributes
- Insert a variable into an existing question
- Create a constant
Arithmetic Operators
The ExamView Test Generator uses two types of operators: arithmetic and relational.

You can use the arithmetic operators to define variables. The ExamView Test Generator uses the following arithmetic operators:

+ Addition (can also concatenate or link two string variables or constants)
- Subtraction
* Multiplication (use 3 * x instead of 3x)
/ Division
\ Integer division
% Modulus
^ Power (use x^2 to represent x to the 2nd power)
! Factorial

Relational Operators
You can use the relational operators to define variables. The ExamView Test Generator uses the following relational operators:

= Equals
< Less than
<= Less than or equal to
> Greater than
>= Greater than or equal to
<> Not equal to
& And (you can also use “and”) | Or (you can also use “or”)

The list of keywords shows the types of variables or constants that can be used as arguments (input) to the various functions, as well as the type of variable that is created by the keyword or function (output).

The ExamView Test Generator also lets you use parentheses to assign precedence to parts of your expressions when the program evaluates them. The following list will help you determine the order in which an expression is evaluated:

( ) parentheses
! + - factorial, positive, negative
^ power
* / \ % multiplication, division, integer division, modulus
+ - addition, subtraction
< <= > >= less than, less than or equals, greater than, greater than or equals
= <> equals, not equals
& and
| or

Define Remaining Question Variables
In this section you will define the remaining variables in question number one.

**Define the number2 Variable**

1. **Open** multiple choice question 1.
2. Click **Edit > Algorithm** Definitions. The Algorithm Definitions window will appear.
3. In the Edit Algorithm Definitions window, click **New**. The New Algorithm window opens.
4. From the Type drop-down list, confirm that **Variable** is selected.
5. Enter **number2** in the Variable name field.
6. Enter **number1+increase** in the Definition field. Do not enter a space before and after the + symbol.
7. From the Format drop-down list, select **Number**.
8. In the Decimal places field, select **2**.
9. Select **Fixed** to show the cents value for the second number in the list.
10. Click **OK**.
11. Click **Done**.

**Insert the number2 Variable**

![Image of the Edit Algorithm window with number2 variable settings]
1. Highlight **5.20** in the question stem.

2. Choose **Insert > Variable**. The Insert Variable window opens.

3. Select **number2** from the list of variable names.

4. Click **Insert**. The program inserts the variable at the current cursor location.

5. Click **Calculate Values** to recalculate a value for number1 and number2. The difference between the first two values is .25.

6. Click **Record** to record the question.

7. Click **Save** to save the question bank.

**Duplicate the number2 Variable**

Algorithms may be duplicated by selecting the variable in the Edit Algorithm Definitions window and pressing CTRL+D (APPLE+D on the Mac). This is a fast way to create new variables.

When you duplicate number2, the new variable will be number3. If you duplicate increase, the new variable will be increase1.

1. Double-click **Multiple Choice 1**.

2. Choose **Edit > Algorithm** Definitions. The Edit Algorithm Definitions window opens.

3. Select **number2**.

4. Press **CTRL+D**. This creates a new variable number3 which is defined exactly the same as number2.

![Edit Algorithm Definitions](image)
**Edit the number3 Variable**

1. With number3 selected, click **Edit**. The Edit Algorithm Definitions window opens.

2. In the Definition field, change `number1` to `number2`. Copying a variable will copy the formatting options as well. Notice the formatting options for number3.

3. Click **OK** to accept the editing changes.

**Duplicate and Edit Remaining Variables**

1. With number3 selected, press **CTRL+D**. Notice this creates a new variable number4 which is defined exactly the same as number3.

2. With number4 selected, click **Edit**. The Edit Algorithm window opens.

3. In the Definition field, change `number2` to `number3`.

4. Click **OK**.

5. With number4 selected, press **CTRL+D**. Notice this creates a new variable number5 which is defined exactly the same as number4.

6. With number5 selected, click **Edit**.

7. In the Definition field, change `number3` to `number4`.

8. Click **OK**.

9. Click **Done**.

**Insert the variables number3, number4, and number5**
1. Highlight 5.40.

2. Choose Insert > Variable. The Insert Variable window opens.

3. Select number3 from the variable name list.

4. Click Insert to insert the variable number3 as the third item in the list.

5. Highlight 5.60.


7. Select number4 from the variable name list.

8. Click Insert to insert the variable number4 as the fourth item in the list.

9. Highlight 5.80.


11. Select number5 from the variable name list.

12. Click Insert to insert the variable number5 as the fifth item in the list.

13. Click Calculate Values to recalculate values for all the variables. Check to be sure all values calculate correctly.

Create Answer Options

All five number variables
You must now create the correct answer, which you will name correct, and the distractors, which you will name wrong1, wrong2, and wrong3.

Distractors are incorrect answers to a Multiple Choice, Bimodal, or Multiple Response question. To create the distractors, you will use logic for mistakes that students commonly make when solving a similar problem. The distractors in our problem will rely on the student adding the tens place incorrectly.

**Create Correct Answer**

1. Choose **Edit > Algorithm** Definitions. The Edit Algorithm Definitions window opens.

2. Select **number5**.

3. Press **CTRL+D**. Notice this creates a new variable number6 which is defined exactly the same as number5.

4. With number6 selected, click **Edit**. The Edit Algorithm window opens.

5. In the Variable name field, enter **correct**.

6. In the Definition field, change **number4** to **number5**.

7. Click **OK**.

**Create Distractors**

Create **wrong1**

1. With correct still selected, press **CTRL+D**. Notice this creates a new variable correct1 which is defined exactly the same as correct.

2. With correct1 selected, click **Edit**.

3. In the Variable name field, enter **wrong1**.

4. In the Definition field, enter **number5+.15**.

5. Click **OK**.

Create **wrong2**
1. With wrong1 still selected, press **CTRL+D**. Notice this creates a new variable wrong2 which is defined exactly the same as wrong1.

2. With wrong2 selected, click **Edit**. The Edit Algorithm window opens.

3. In the Definition field, enter **number5+.35**.

4. Click **OK**.

Create **wrong3**

1. With wrong2 still highlighted, press **CTRL+D**. Notice this creates a new variable wrong3 which is defined exactly the same as wrong2.

2. With wrong3 selected, click **Edit**. The Edit Algorithms window opens.

3. In the Definition field, enter **number5+.05**.

4. Click **OK** to accept the edits.

5. Click **Done**.

---

**Insert the variables correct, wrong1, wrong2, and wrong3**
1. Highlight 6.00 in answer choice c.

2. Choose Insert > Variable. The Insert Variable window opens.

3. Select correct in the list of variables.

4. Click Insert.


7. Select wrong1 in the list of variables.

8. Click Insert.

9. Highlight 5.95 in answer choice b.


11. Select wrong2 in the list of variables.

12. Click Insert.

13. Highlight 6.35 in answer choice d.


15. Select wrong3 in the list of variables.

16. Click Insert to insert the last distracter into the answer options.

17. Click Calculate Values to recalculate values for all variables.

18. Click Record to record the question. The calculator that appears to the left of the question number to indicate the question is now dynamic.

19. Click Save to save the question bank.
**Format the Coefficients**

Formatting variables as Coefficient or Coefficient (1st) will only be used in algebraic equations. *If you do not plan to include algebraic equations in your question banks, you may skip this section.*

- The Coefficient (1st) format will hide a leading 1 or -1.
- The Coefficient format will hide a 1 or -1. If the number substituted for the variable is negative, a minus sign will be displayed. Otherwise, a plus sign will be displayed.

**Create Coefficient “vara”**

You will make the expression $4x + 3 = 11$ into $vara \times varb = varc$ making vara, varb, and varc into variables. You will format vara as Coefficient (1st) to hide the 1 if the value of the variable is 1 or -1.

1. **Double-click** Multiple Choice 2. The edit window opens.
2. Choose **Edit > Algorithm Definitions**. The Edit Algorithm Definitions window opens.
3. Click **New**. The New Algorithm window opens. From the Type drop-down list, confirm that Variable is selected.
4. Enter **vara** in the Variable name field.
5. Enter **range(-9,9,1)** in the Definition field. The definition `range(-9,9,1)` will choose a random number between -9 and 9 incrementing by 1 and skipping 0.
6. From the Format drop-down list, select Coefficient (1st).
7. Click **OK**.

**Create Coefficient “varb”**

You will format varb as Coefficient to hide the value if it is a positive or negative 1 and insert the correct operation symbol before the term.

1. Click **New**. The New Algorithm window opens.
2. Enter **varb** in the Variable name field.
3. Enter **range(-9,9,1)** in the Definition field.
4. In the Format drop-down list, select **Coefficient**.
5. Click **OK**.
6. Click **Done**.
Insert Coefficient Variables

1. Highlight the **number 4** in the question.

2. Choose **Insert > Variable**. The Insert Variable window opens.

3. With **vara** selected, click **Insert**.

4. Highlight the **number 3** in the question.

5. Choose **Insert > Variable**. The Insert Variable window opens.

6. Select **varb** from the variable list.

7. Click **Insert**.

8. Test the variables using **Calculate Values**. You will immediately notice the double signs (+ – or ++). Remember that formatting the second variable as coefficient will generate a plus or minus based on the sign of varb.

9. **Delete** the + sign and the extra space preceding it.

   Solve for x
   \[ varax \ \text{varb} = 11 \]
   \[ \begin{array}{ccc}
   \text{a} & 5 \\
   \text{b} & 7 \\
   \end{array} \]

10. Click **Calculate Values** to recalculate values for the variables. Continue clicking Calculate Values at least 15 times. Eventually varb will display just a negative (-) sign. To fix this follow the steps in the next section.

    Solve for x.
    \[ 9x - 11 \]
    \[ \begin{array}{ccc}
    \text{a} & 5 \\
    \text{b} & 7 \\
    \end{array} \]

    “varb” incorrectly displaying a negative sign
Display \textit{varb} Correctly
Since \textit{varb} is not a coefficient but a constant, you will want to use the functions \texttt{abs}, which provides the absolute value of the number, and \texttt{sgns}, which returns the sign of the number to display \textit{varb}.


2. Click \textbf{New}. The New Algorithm window opens.

3. Enter \texttt{absvarb} in the Variable name field. The abs function returns the absolute value of the variable.

4. Enter \texttt{abs(varb)} in the Definition field to return the absolute value of variable \textit{varb}.

5. Confirm that \textbf{Format} is set to \textbf{General}.

6. Click \textbf{OK}.

\begin{center}
\includegraphics[width=0.8\textwidth]{create_absvarb_variable.png}
\end{center}

Create “\textit{absvarb}” variable

Configure Sign of \textit{varb}

1. Click \textbf{New}. The New Algorithm window opens. Confirm that \textbf{Type} is set to \textbf{Variable}.

2. Enter \texttt{signvarb} in the \textbf{Variable} name field.

3. Enter \texttt{sgns(varb)} in the \textbf{Definition} field. The sgns function returns the sign (+ or –) of the variable.

4. Confirm that \textbf{Format} is set to \textbf{General}.

5. Click \textbf{OK}.

6. Click \textbf{Done}.
Replace varb with signvarb and absvarb

1. **Delete** everything between the x and the = in the question.

2. Position the cursor immediately after the x. Press the **SPACEBAR** one time.

3. Choose **Insert > Variable**. The Insert Variable window opens.

4. Select **signvarb** from the variable list.

5. Click **Insert**.

6. Press the **SPACEBAR** one time.

7. Choose **Insert > Variable**. The Insert Variable window opens.

8. Select **absvarb** from the variable list.

9. Click **Insert**.

10. Press the **SPACEBAR** one time.

<table>
<thead>
<tr>
<th>Solve for x</th>
<th>varb: signvarb absvarb = 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. 5</td>
<td>c. 2</td>
</tr>
<tr>
<td>b. 7</td>
<td>d. 3</td>
</tr>
</tbody>
</table>

11. Click **Calculate Values** to recalculate values for all variables at least 5 times.
Create the Correct Answer Variable
If you prefer that your answer is an integer, you should define your answer as the equation \( vara \times correct + varb = varc \).

1. Choose **Edit > Algorithm** Definitions. The Edit Algorithm Definitions window opens.

2. Click **New**. The New Algorithm window opens.

3. Enter **correct** in the **Variable name** field.

4. Enter **range(-9,9)** in the **Definition** field.

5. Confirm that **Format** is set to **General**.

6. Click **OK**.

7. Click **Done**.

Insert the Correct Answer Variable

1. **Highlight** the number 2 in answer choice c.

2. Choose **Insert > Variable**. The Insert Variable window opens.

3. Select **correct** from the Insert Variable list.

4. Click **Insert**.

```latex
\text{Solve for } x. \\
vara, \text{sign}varb, \text{abs}varb = 11 \\
\text{a. } 5 \\
\text{b. } 7 \\
\text{c. } \text{correct} \\
\text{d. } 3
```
Create varc
Create varc to complete the equation \( vara*correct+varb = varc \)

1. Choose **Edit > Algorithm Definitions.** The Edit Algorithm Definitions window opens.
2. Click **New.** The New Algorithm window opens.
3. Enter **varc** in the **Variable name** field.
4. Enter **vara*correct+varb** in the **Definition** field.
5. Confirm that **Format** is set to **General.**
6. Click **OK.**
7. Click **Done.**

Insert varc into the question.

1. **Highlight** the number 11 in the question.
2. Choose **Insert > Variable.** The Insert Variable window opens.
3. Select **varc** from the Insert variable list.
4. Click **Insert** to insert varc into the question stem.

5. Click **Calculate Values** to recalculate values for all variables. Do this at least five times. You will want to check to see that the answer is correct given the equation that appears.
Use Format Functions

It is now important to create your distractors (incorrect answers). A common mistake in solving algebraic equations is to add the same number to each side of the equation instead of adding the opposite to each side of the equation. If this is done some of the answer options will be non-integer numbers.

To format the non-integer wrong answers, you will use the fraction formatting functions that are provided in the algorithmic functions. The ExamView Test Generator provides numerous fraction functions to use when defining variables, conditions, and user-defined functions.

- The fraction formatting functions include fracs, mixfracs, rfracs, sfracs, smixfracs, and srfracs.
  - Fracs returns a stacked fraction in reduced form.
  - Mixfracs returns a mixed fraction in reduced form.
  - Rfracs returns a fraction in reduced ratio form.
  - Sfracs returns a small stacked fraction in reduced form.
  - Smixfracs returns a small mixed fraction in reduced form.
  - Srfracs returns a small fraction in reduced ratio form.
- Variables defined using these formatting functions are for display only and cannot be used as parts of an expression.
- Other formatting functions that are used for display only include decs (displays a repeating decimal), sciens (displays a number in scientific notation), and sqrs (displays a radical in rational form).
Create \textit{wrong1}

1. Choose \textbf{Edit} > \textbf{Algorithm} Definitions. The Edit Algorithm Definitions window opens.

2. Click \textbf{New}. The New Algorithm window opens.

3. Enter \textit{wrong1} in the Variable name field.

4. Enter \text{
\texttt{fracs(varc+varb,vara)}}\textit{ in the Definition field.}

- With the fracs formatting function, the first parameter defines the numerator and the second parameter is the denominator of the fraction.
- All fractions will be displayed in reduced form. The rfracs and srfracs fractions will appear in reduced ratio form.

5. Confirm that the \textit{format} is set to \textbf{General}.

6. Click \textbf{OK} and click \textbf{Done}.

Insert \textit{wrong1}

1. \textbf{Delete} the answer in answer choice a.

2. With the cursor positioned in answer choice a, choose \textbf{Insert} > \textbf{Variable}.

3. Select \textit{wrong1} from the Insert Variable list.

4. Click \textbf{Insert}.

5. Click \textbf{Calculate Values} \text{CALC} to recalculate values for all variables. Do this at least three times.
**Create wrong2**

To eliminate the possibility of a guess answer, you will want to create something close to the correct answer. You will create the variable wrong2 to eliminate the possibility of a guess. It will be created as a number within two of the correct answer.

1. Choose **Edit > Algorithm** Definitions. The Edit Algorithm Definitions window opens.

2. Click **New**. The New Algorithm window opens.

3. Enter **wrong2** in the **Variable name** field.

4. Enter **correct+range(-2,2,1)** in the **Definition field**. Adding a range of positive and negative values adds more variability to the problem.

5. Click **OK**.

6. Click **Done**.

**Insert wrong2**

1. **Delete** the answer in answer choice b.

2. With the cursor positioned in answer choice b, choose **Insert > Variable**.

3. Select **wrong2** from the Insert Variable list.

4. Click **Insert**.
Create \textit{wrong3}
You will create \textit{wrong3} to be a number within two of \textit{wrong1}. Since \textit{wrong1} is defined by fracs, it becomes a string variable. You will therefore not be able to use \textit{wrong1} in a numeric calculation.

1. Choose \textit{Edit} > \textit{Algorithm} Definitions. The Edit Algorithm Definitions window opens.
2. Click \texttt{New}.
3. Enter \texttt{wrong3} in the \texttt{Variable name} field.
4. Enter \texttt{range(-2,2,1)+((varc+varb)/vara)} in the \texttt{Definition} field.
5. Click \texttt{OK}.

Format \textit{wrong3} as \textit{wrong3display}
The variable \textit{wrong3display} will display \textit{wrong3} as a repeating decimal.

1. Click \texttt{New}. The New Algorithm window opens.
2. Enter \texttt{wrong3display} in the \texttt{Variable name} field.
3. Enter \texttt{decs(wrong3)} in the \texttt{Definition} field.
4. Click \texttt{OK}.
5. Click \texttt{Done}.

Insert \textit{wrong3display}

1. \textbf{Delete} the answer in answer choice d.
2. With the cursor positioned in answer choice d, choose \textit{Insert} > \textit{Variable}.
3. Select \texttt{wrong3display} from the Insert Variable list.
4. Click \texttt{Insert}.
5. Click \texttt{Calculate Values} at least 15 times to recalculate values for all variables.
Define Conditions
You may notice while recalculating that occasionally you will see two of the same answers appear in separate answer choices.

Solve for \( x \).

\[
6x + 3 = 39
\]

\[
\begin{array}{ll}
\text{a.} & -7 \\
\text{b.} & -7 \\
\text{c.} & -6 \\
\text{d.} & -8 \\
\end{array}
\]

To eliminate this from occurring, you will use the \textit{isunique} condition. Conditions do not need a name. All conditions entered for an algorithm group must evaluate to TRUE when the item is recorded or when Calculate Values is chosen. If a condition evaluates to FALSE, the program will try to evaluate the algorithms again using different values for random variables. If the conditions are not able to be satisfied within a certain amount of time, the program will inform you that the conditions could not be met.

It is always best to keep the number of conditions to a minimum. This makes the software run more quickly and efficiently.

1. Choose \textit{Edit} > \textit{Algorithm} Definitions. The Edit Algorithm Definitions window opens.
2. Click \textit{New}. The New Algorithm window opens.
3. From the \textit{Type} drop-down list, select \textit{Condition}.
4. In the \textit{condition} field, enter

\[
\text{isunique}(\text{correct}, (\text{varc}+\text{varb})/\text{vara}, \text{wrong2}, \text{wrong3})
\]

\textit{Note:} That all parameters in the \textit{isunique} statements must be of the same type. You may not mix string values and number values. Therefore, you cannot use the variable \textit{wrong1} as one of the parameters since it is a string and the others are numbers.

5. Click \textit{OK}.
6. Click \textit{Done}.
7. Click \textit{Calculate Values} at least 15 times, checking for duplicate values in the answer choices.
Use Scramble True
Entering Scramble as the variable name and True as the definition will cause the answer choices in Multiple Choice and Multiple Response questions to change places when the question is selected or chosen and Calculate Values is selected.


2. Click New. The New Algorithm window opens.

3. Enter scramble in the Variable name field.

4. Enter true in the Definition field.

5. Click OK.

6. Click Done.

7. Click Calculate Values to recalculate values for all variables. Do this five times. Notice that the correct answer changes location within the answer choices.

8. Click Record to record the question.

9. Click Save to save the question bank.
Use Lists

Lists are used to hold a group of similar expressions.

- Expressions must all be of the same type, either numeric or string.
- If you create a list of strings you must be sure that no string element contains a comma, since a comma is used to delimit a list.
- A variable defined as a list can be used as the argument for a choose function.

When the moon in Multiple Choice 3 is at point A, it is a new moon, point B is a waxing crescent, point C is a first quarter moon, point D is a waxing gibbous moon, and point E is a full moon. You will associate the point on the diagram with the correct moon phase.

Use Rand Function

1. Double-click Multiple Choice 3.

2. Choose Edit > Algorithm Definitions. The Edit Algorithm Definitions window opens.

3. Click New to create a new variable.

4. Enter pointer in the Variable name field.

5. Enter rand(5) in the Definition field. Rand(x) is a function that returns a random whole number between one and x. Since there will be 5 phases of the moon listed, enter 5 for the parameter x.

6. Confirm that Format is set to General.

7. Click OK.

Create letterlist list

1. Click New.

2. Enter letterlist in the Variable name field.

3. Enter list("A","B","C","D","E") in the Definition field.

4. Click OK.
Create `correctlist` list
Use `correctlist` for the list of the 5 phases of the moon shown on the diagram. It is important to position the phases of the moon in `correctlist` in the exact same position in the list as its corresponding point in `letterlist`.

1. Click **New**.
2. Enter `correctlist` in the **Variable name** field.
3. Enter \texttt{list("new moon", "waxing crescent", "first quarter", "waxing gibbous", "full moon")} in the **Definition** field.
4. Click **OK**.

Use Choose Function

1. Click **New**.
2. Enter `letter` in the **Variable name** field. The variable `letter` will indicate the point in the moon's orbit.
3. Enter \texttt{choose(pointer,letterlist)} in the **Definition** field. The choose function will select the argument at the pointer location in the list `letterlist`.
4. Click **OK**.
5. Click **New**.
6. Enter `correct` in the **Variable name** field. The variable `correct` will indicate the name of the correct moon phase.
7. Enter \texttt{choose(pointer,correctlist)} in the **Definition** field. This choose function will select the argument at the pointer location in the list `correctlist`.
8. Click **OK**.
9. Click **Done**.
Insert the Variables

1. **Highlight** B in the question stem, and choose **Insert > Variable**.
2. Select **letter** from the Insert Variable list.
3. Click **Insert**.
4. **Highlight** choice b in the answer choices, and choose **Insert | Variable**.
5. Select **correct** from the Insert Variable list.
6. Click **Insert**.

Delete the Display Variable

You may wish to display the variable pointer at the end of the question as a way to check your work. You would then be able to see whether the correct value from each list is selected based on the value of pointer. After recalculating several times, you must delete the display of this value.

1. Place the cursor at the end of the question, and choose **Insert > Variable**.
2. Select **pointer** from the **Insert Variable** list.
3. Click **Insert**.
4. Click **Calculate Values** to recalculate values for all variables. Do this 10 times. Check to be sure that the correct point and moon phase are displayed based on the value of pointer.
5. Since the values appear to be correct, you must delete the display of the value pointer. **Highlight** the value of pointer at the end of the question and press the **DELETE** key on your keyboard.
Add Distractors
To avoid creating a condition, you will build your distractors in such a way that no duplicates could result.

1. Choose **Edit > Algorithm** Definitions. The Edit Algorithm Definitions window opens.
2. Click **New**.
3. Enter **wrong1** in the **Variable name** field.
4. Enter **choose(pointer+1, correctlist)** in the **Definition** field.
5. Click **OK**.
6. Click **New**.
7. Enter **wrong2** in the **Variable name** field.
8. Enter **choose(pointer+2, correctlist)** in the **Definition** field.
9. Click **OK**.
10. Click **New**.
11. Enter **wrong3** in the **Variable name** field.
12. Enter **choose(pointer+3, correctlist)** in the **Definition** field.
13. Click **OK**.

Scramble Answer Options
Add the Scramble True option to the question so that the answer choices will scramble when the question is recalculated.

1. Click **New**. The New Algorithm window opens.
2. Enter **scramble** in the **Variable name** field.
3. Enter **true** in the **Definition** field.
4. Click **OK**.
5. Click **Done**.
Insert the variables into the answer choices

1. Insert **wrong1** in answer choice a, **wrong2** in answer choice c, and **wrong3** in answer choice d.

2. Click **Calculate Values** at least five times to check your work.

3. Click **Record** to record the question.

4. Click **Save** to save the question bank.

The diagram shows the Moon at eight points in its orbit around the Earth. What moon phase is seen on Earth when the Moon is at point letter in its orbit?

- **a**. **wrong2**
- **b**. **wrong3**
- **c**. **correct**
- **d**. **wrong1**
Use the Number Line

There are three graphing options in ExamView: Number Line, Polar, and Cartesian. In this session, you will learn about the number line and the Cartesian graph. The Number Line feature includes options to add points, segments, rays, lines, box-and-whiskers plots, and vectors.

Add Points to the Number Line

For the steps in this section, you will format coefficients in Short Answer 1.

1. **Double-click** Short Answer 1. The Short Answer question window opens.
   - You will replace the enhanced metafile with a number line created in the software and add a point that varies.

2. **Delete** the number line from the question stem and the answer.

3. Place the cursor in the answer field. Choose **Insert > Graph > Number Line**.
   - A number line with default properties will be inserted, and the Format Graph – Number Line dialog will be displayed.
   - There are three tabs in this dialog that allow you to format the number line in different ways.
   - The Functions tab lets you add, edit, and delete items from the number line. These items include points, segments, rays, lines, box-and-whisker plots, and vectors.

4. Confirm that Point is selected from the drop-down list and **click New**. The New Point window opens.

5. Enter 25 in the **Coordinate** box.

6. **Click OK**.

7. **Select the View tab**.
   - To see the point you must change the view.
   - The View tab lets you change the size of the number line and set the number line centered on a certain value or anchored at the left of a certain value.
8. Verify that **Set Center** is selected. **Enter 24** for the coordinate. This will center the graph at 24.

9. Click **Apply**. Note the changes to the number line.

10. **Select** the **Axis** tab.

   - The Axis tab lets you change attributes of the number line, such as the scale (units per tick), ticks per inch, and line styles.
   - With the Axis tab displayed, you may enter a label to display at the right of the number line or choose to leave the label blank. You will leave the label blank.

11. Change the **Scale** (units per tick) to **2** and click **Apply**. Note the changes to the number line. The labels are now incremented by 2.

12. Change the **Ticks per inch** to **3** and click **Apply**. Note the changes to the number line. The lower the number of ticks per inch the more room between ticks on the number line.

13. Change the **Ticks per label** to **2** and click **Apply**. Note the changes to the number line. Every other tick is labeled.

   - Other options in the Axis tab include setting the axis width and color, showing or hiding tick marks and tick mark labels, and zooming in or out.
   - To enter alternate text to be displayed when the student points to the graph while taking an online test, you would choose the Web tab.

14. Click **OK**.

15. **Click** on the number line. **Drag the right border** until the number line begins at **12** and ends at **36**.

   - If you inadvertently apply changes that you do not want to the number line, close the Format Graph – Number Line dialog and choose **Edit > Undo**. This will cancel all changes made while the dialog was displayed, including inserting new items, so be careful when choosing this option.
Create Blank Number Line
You will now need to provide a blank number line for the student to use when answering the question on the test. Do this by copying the number line into the question stem and removing the answer.

1. **Click** on the number line and choose **Edit > Copy**.

2. Place your cursor at the end of the question and press **ENTER**.

3. Choose **Edit > Paste** to paste the number line into the question.

4. **Double-click** the number line in the question. The Format Graph - Number Line dialog box opens.

5. **Delete** the **Point: 25 entry** by highlighting and clicking Delete.

6. Choose **Yes** when asked if you would like to delete this graph function.

7. Click **OK**.

8. Click **Record** to record the question. Then, click **Save** to save the question bank.
Add Variables to the Number Line

Variables cannot be used as values for the number line label, units per tick, ticks per inch, ticks per label, width, height, or baseline. Variables can be used to define most of the attributes for the items added to the number line.

Create and Insert a Point Variable

1. Select Short Answer 1 and click Edit. The Short Answer question window opens.
2. Choose Edit > Algorithm Definitions. The Edit Algorithm Definitions window opens.
3. Click New.
4. Enter number in the Variable name field.
5. Enter range(12,34,3) in the Definition field.
6. Confirm that Format is set to General.
7. Click OK.
8. Click Done.
9. Highlight the number 25 in the question stem, and choose Insert > Variable.
10. Select number from the Insert Variable list.
11. Click Insert.
12. Double-click on the number line in the answer. The Format Graph – Number Line window opens.
14. Replace 25 with the word number in the Coordinate box.
15. Click OK twice.
16. Click Calculate Values at least five times to check your work. Be sure the question and the answer match.
17. Click Record to record the question. Then, click Save to save the question bank.
Add Rays to the Number Line

A ray can be used to show that a variable is less than, less than or equal to, greater than, or greater than or equal to a certain value.

1. Select Short Answer 1 and choose Question > Duplicate. Short Answer 2 is created.

2. Select Short Answer 2 and click Edit.

3. Change = to < in the question stem.

4. Double-click on the graph in the answer. The Format Graph – Number Line window opens.

5. With Point: number selected, click Delete.

6. Choose Yes when asked if you would like to delete this graph function.

7. Select Ray from the function drop down menu, and click New. The New Ray dialog box opens.

8. Confirm that Direction is set to Left. The default direction for ray is to the left. Since you are graphing “less than”, you will want the ray pointing left.

9. Enter number in the Coordinate field.

10. Select the open circle for Point style since the question asks “less than” and not “less than or equal to”.

11. In the Color drop-down box, choose Blue.

12. Click Apply to apply the changes.

13. Click OK twice.

14. Click Calculate Values at least five times to check your work.

15. Click Record to record the question. Then, click Save to save the question bank.
Use the Display only if Box
You may display rays pointing to either the right or left on the same graph using the “Display ray only if” box.

You will edit Short Answer 3 to become a question that includes options: greater than, less than, greater than or equal to, or less than or equal to.

1. Select Short Answer 2 and choose **Question > Duplicate**. Short Answer 3 is created.

2. Select Short Answer 3 and click **Edit**. Create a variable named which that will be used as a pointer to point at one of four options: greater than, less than, greater than or equal to, or less than or equal to.

3. Choose **Edit > Algorithm Definitions**. The Edit Algorithm Definitions window opens.

4. Click **New**.

5. Enter **which** in the **Variable name** field.

6. Enter **rand(4)** in the **Definition** field.

7. Click **OK**.

8. Click **Done**.

You will now choose to display less than when which is 1, greater when which is 2, less than or equal to when which is 3, and than or equal to when which is 4.

9. **Double-click** on the number line in the answer. The Format Graph – Number Line window opens.

10. With **Ray:number** to the left **highlighted**, click **Edit**.

11. In the **Display ray(only if…)** box, enter **which = 1**. Be sure the box preceding this line is checked.

12. Confirm that the **Point style** is the **open circle**.

13. Click **OK**.

Duplicate and edit the first function in the Format Graph – Line window.

14. With the Format Graph – Number Line window open, press **CTRL+D (APPLE+D on the Mac)**.

15. Change the **Display ray(only if ...)** to **which = 2**.
16. Confirm the **Point style** is the open circle.

17. Click **OK**.

Duplicate and edit the second function in the Format Graph – Number Line window.

18. With the Format Graph – Number Line window open, press **CTRL+D (APPLE+D on the Mac)**.

Edit this new ray to display the ray representing “less than or equal to” when which = 3.

19. Click **Edit**.

20. Change the **Direction** to Left.

21. Change the **Point style** to the closed circle.

22. Change the **Display ray(only if ...)** to which = 3.

23. Click **OK**.

Duplicate and edit the third function in the Format Graph – Number Line window.

24. With the Format Graph – Number Line window open, press **CTRL+D (APPLE+D on the Mac)**.

Edit this new ray to display the ray representing “greater than or equal to” when which = 4.

25. Click **Edit**.

26. Change the **Direction** to Right.

27. Confirm that the **Point style** is the closed circle.

28. Change the **Display ray(only if ...)** to which = 4.

29. Click **OK twice** to close the Edit Ray window and the Format Graph – Number Line window.
You must now change the relational symbol to show “<” when which = 1, “>” when which = 2, “≤” when which = 3, and “≥” when which = 4.


31. Click New.

32. Enter relation in the Variable name field.
   - Symbol(x) converts the string argument x to the appropriate symbol character.
   - Once the variable is inserted into a question or narrative, you must highlight the variable and change it to Symbol font for it to display properly.
   - The following list shows valid arguments to this function.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Will return</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=</td>
<td>≤</td>
</tr>
<tr>
<td>&gt;=</td>
<td>≥</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>≠</td>
</tr>
<tr>
<td>x</td>
<td>×</td>
</tr>
<tr>
<td>.</td>
<td>⋅</td>
</tr>
<tr>
<td>/</td>
<td>÷</td>
</tr>
</tbody>
</table>

   - For all other Symbol characters, use the chr function.

33. Enter

\[
\text{choose(\text{which,","","","\text{symbol("<=")},\text{symbol("\ge")}}\})
\]

in the Definition field.

34. Click OK.

35. Click Done.

Insert the variable relation into the question stem.

Relational symbols
36. **Highlight** the `< sign` in the question stem. Choose **Insert > Variable**.

37. Choose the **variable relation** and click **Insert**.
Change relation to Symbol font. If left in Times New Roman font, the ≤ and ≥ symbols will not display properly.

38. **Highlight relation** and change the **font** to **Symbol**.

Check your work.

39. Click **Calculate Values** at least 10 times to check your work.

40. Click **Record** to record the question.

41. Click **Save** to save the question bank.

*Inserting the “relation” variable*
Use the Functions on the Cartesian Graph

A Cartesian graph is a graph defined by Cartesian or rectangular coordinates $x$ (the horizontal coordinate) and $y$ (the vertical coordinate). The horizontal coordinate ($x$) is the horizontal distance from a point to the origin. The vertical coordinate ($y$) is the vertical distance from a point to the origin.

Use the Functions $f(x) =$

For the steps in this section, you will use the $f(x)$ function in Problem 1.

1. **Select** Problem 1 and click **Edit**.

2. Click on the graph and press **DELETE** on your keyboard.

3. Choose **Insert > Graph > Cartesian**. A Cartesian graph with default properties will be inserted, and the Format Graph – Cartesian window will be displayed.

Use the Functions Tab

There are three tabs in this dialog that allow you to format the graph in different ways: the functions tab, the axes tab, and the view tab. Let’s first cover the functions tab.

The Functions tab lets you add, edit, and delete items from the graph. These items include $f(x)$, $f(y)$, Parabola, Circle, Ellipse, Hyperbola, Point($x$, $y$), Segment, Polynomial, Parametric, Normal Curve, Picture, Shape, Pie Chart, Text Box, and Compound.

Use the $f(x)$ function to graph the line $y = 3x + 1$ on the graph.

1. Choose the **$f(x)$ function** in the **Function drop-down menu** and click **New**.

2. Next to $y =$ enter $3*x+1$. You must use the * to indicate multiplication in the function.

3. Change the **Color** to **Blue**.

4. Click **Apply**.

5. Click **OK**.

---

Introduction to Building Dynamic Content
Use the **Axes Tab**
The Axes tab lets you change attributes of the horizontal and vertical axes, such as the scale (units per tick), ticks per inch, and line styles.

1. Choose the **Axes tab**.

2. Change the **ticks per inch** to 6 for both the vertical and horizontal axis and click **Apply**. Note the decreased distance between labels.

3. Change the **ticks per label** to 2 for both the vertical and horizontal axis and click **Apply**. Note that only every other label is displayed.

Use the **View Tab**
The View tab lets you change the size and background of the graph and set the graph centered on a certain value or anchored at the lower left to a certain value.

1. Choose the **View tab**.

2. Change the **Background** to **Lined**. The background can be blank, dotted, or lined. By default, the background is blank. Lined or dotted backgrounds help define the coordinate system.

3. Change the **Background color** to **Dark Blue**.

4. Choose **Apply**. Note the changes to the graph.

Use the **Web Tab**
With the Web tab displayed, you may choose to enter alternate text to be displayed when the student points to the graph while taking an online test. This option is to aid the visually impaired.

5. Click **OK** to record your changes and close the Format Graph - Cartesian dialog.
**Make the Graph Dynamic**

For the steps in this section, you will make the graph in Problem 1 dynamic. Both the slope and the y-intercept may become variables in this problem.

1. Choose **Edit > Algorithm Definitions**. The Edit Algorithm Definitions window opens.

2. Click **New**.

Define the variable slope as a number between -5 and 5 and not equal to zero.

3. Enter **slope** in the **Variable name** field.

4. Enter **range(-5,5,1)** in the **Definition** field and click **OK**. Note that including the third parameter also tells the software to skip zero.

Create the variable intercept to represent the position on the graph where the line will cross the y-axis.

5. Click **New**.

6. Enter **intercept** in the **Variable name** field.

7. Enter **range(-5,5)** in the **Definition** field. Note that by excluding the third parameter, you allow the variable intercept to be equal to zero.

8. Click **OK**.

9. Click **Done**.

You must now insert the variables into the equation of the f(x) function.

10. **Double-click** on the **Cartesian graph**. The Format Graph – Cartesian dialog box opens.

11. Confirm that the **function y = 3*x + 1 highlighted** and click **Edit**.

12. Replace **3*x + 1** with **slope*x + intercept**.

13. Click **Apply**.

14. Click **OK** twice.
You must now insert the variable *slope* into the answer.

15. **Delete 3** from the answer.

16. Making sure the cursor is still in the answer, choose **Insert > Variable**.

17. Select *slope* and click **Insert**. The slope variable is now inserted into the answer.

18. Click **Calculate Values** at least five times to check your work.

19. Click **Record** to record the question.

20. Click **Save** to save the question bank.
Make the Graph Dynamic
For the steps in this section, you will make the graph in Problem 1 dynamic. Both the slope and the y-

Use the Circle Function
To define and create a circle, you will need a radius and a center point (if the circle will be centered at a location other than the origin). For the steps in this section, you will use the Circle function to change Problem 2 to a dynamic question.

1. Select Problem 2 and click Edit.

2. Choose Edit > Algorithm Definitions. The Edit Algorithm Definitions window opens.

3. Click New.

4. You will first define a variable radius for the radius of the circle. Enter radius in the Variable name field.

5. Enter range(1,4) in the Definition field.

6. Click OK.

Define the variables for the x and y coordinates of the center point.

7. Click New.

8. Enter xvalue in the Variable name field.

9. Enter range (-3,3) in the Definition field.

10. Click OK.

11. Enter yvalue in the Variable name field.

12. Enter range (-3,3) in the Definition field.

13. Click OK.

14. Click Done.

You will use these variables when defining your circle. You must delete the graph presently in Problem 2 and replace it with the Cartesian graph made by the software.

15. Highlight the picture of the graph, and press delete on your keyboard.

16. With the cursor on the line below the question, choose Insert > Graph > Cartesian. The Format Graph - Cartesian window opens.
17. With the Functions tab displayed, click on the drop-down list and select Circle.

18. Click the New button. The New Circle dialog box opens.

19. Choose a circle type. The circle can be centered on the origin or on an arbitrary point. Choose the option \((x - h)2 + (y - k)2 = r2\).

Choose a relation: equals, less than, less than or equal to, greater than, or greater than or equal to. The relation chosen determines the line style for the circle as well as whether the area inside or outside the circle is shaded.

20. Confirm that the relation is set to =.

21. Enter radius in the r field.

22. If the circle is not centered on the origin, enter values for the center \((h, k)\). Enter the variables for \(h\) (x-coordinate) and \(k\) (y-coordinate). Enter \(xvalue\) in the \(h\) field and \(yvalue\) in the \(k\) field.

- If you want the circle to display only if a certain condition is met, enter the condition in the field following Display circle (only if ...). You will not enter a condition.

- You may click the Pattern pop-up menu to change the drawing style for the circle. Your pattern will remain the solid line.

- You may click the Color pop-up menu to change the color of the circle. Your color will remain black.

- You may click the Shading pop-up menu to change the style for the shaded region. If the relation is “=”, the shading style is not used. You will not use the shading.

23. Click OK.
To place lines on the background of the graph, you will need to make changes to the View tab.

24. Choose the **View tab**.

25. Choose **Lined** for the **Background**.

26. Choose **Dark Blue** for the **Line Color**.

To allow for enough space to show the circle on the graph, you can change the size of the graph by either dragging one of the corners of the graph or making changes to the size in the View tab.

27. Change the **Width, Height** and **Baseline** to **3.5”**.

28. Click **Apply** and then **OK**.

You will not see the circle on the graph until you click **Calculate Values**. Until you recalculate, the software is unable to determine the values of the variables.

29. Click **Calculate Values** to view the circle.
The formula for the area of the circle is radius squared times pi. You must define a variable for the radius squared to insert into the answer.


31. Click New.

32. Enter radiussq in the Variable name field.

Use the ^ key to indicate an exponent will follow. For example, use x^2 to represent x to the 2nd power.

33. Enter radius^2 in the Definition field.

34. Click OK.

35. Click Done.

Insert the variable radiussq into the answer.

36. Highlight the 4 in the answer.

37. Choose Insert > Variable.

38. Choose radiussq and click Insert.

39. Click Calculate Values at least five times to check your work.

40. Click Record to record the question.

41. Click Save to save the question bank.
Use the Picture Function

- You can add picture files, such as bitmap and jpegs, to a Cartesian graph.
- For the steps in this section, you will use Completion 1 to create a question with dynamic pictures.
- You will make the question display five different states, changing the answer to match the state shown.

**Step 1:** Select **Completion 1** and click **Edit**.

**Step 2:** Click the picture of the state of Ohio and press **DELETE** on your keyboard.

- You will need to create a variable, *pointer*, to point at one of five random choices of states listed in the choose function.

**Step 3:** Choose **Edit | Algorithm Definitions**. The Edit Algorithm Definitions window opens.

**Step 4:** Click **New**.

**Step 5:** Enter **pointer** in the Variable name field.

**Step 6:** Enter **rand(5)** in the Definition field.

**Step 7:** Click **OK**.

- You will create a variable for the state name. The variable *statename* will use the value of the variable *pointer* to choose the state.

**Step 8:** Click **New**.

**Step 9:** Enter **statename** in the Variable name field.

**Step 10:** Enter **choose(pointer,"Ohio","Kansas","Texas","Michigan","Florida")** in the Definition field. (See Figure 52.)

**Step 11:** Click **OK**.

![Edit Algorithm Definitions - States](image)

**Figure 52** Edit Algorithm Definitions - States

**Step 12:** Click **Done**.
• Insert the answer in the answer field.

**Step 13:** Delete the word Ohio from the answer field.

**Step 14:** With the cursor in the answer field, choose **Insert | Variable**.

**Step 15:** Choose the variable `statename` and click **Insert**.

**Step 16:** Click **Calculate Values** at least five times to watch the names of the states change.

• You now need to add the pictures of the states. You will do this in the Cartesian graph using the Display (only if...) option to help you display the correct picture based on the value of the variable `pointer`.

**Step 17:** With the cursor on the line below the question, choose **Insert | Graph | Cartesian**.

• Select the **Picture** function.

**Step 18:** With the Functions tab displayed, click on the drop-down list and select **Picture**.

**Step 19:** Click **New**.

• Browse to find the picture of the state of Ohio.

**Step 20:** Click **Select** and browse to find the file `OHshape.bmp` located in the ExamView | Banks | Training | state bitmaps folder.

• The default picture type is bitmap (.bmp). If you would need a file of another type, click the List files of type box and choose the type of picture file you want to insert.

• The program supports the following formats: bitmap (.bmp), jpeg (.jpg), TIFF (.tif), and Macintosh PICT (.pct).

• Pictures inserted into a graph are required to be monochrome (black and white) or 16 color (default Windows palette). If you insert a picture that requires more colors, the program will reduce the colors 16. It is recommended that only black and white pictures be added to graphs.

**Step 21:** After locating the `OHshape.bmp` file, double-click on the file. The bitmap (or part of the bitmap) will appear in the Preview window.

**Step 22:** Enter **Ohio** in the Picture name field.

• You may need to change the size of the picture to make it fit and center it in the preview window.

• Variables can be used for the picture name, or as values for left x, top y, width, and height.

**Step 23:** Enter **10** for the Height and **5** for the Top Y. Using half the height for the top y value will center the image vertically.

• Note that since the **Lock Aspect Ratio** is checked, the width changes proportionally with the change in height.

• You will however need to center the picture horizontally. Use the opposite of half the value of the width to center the image horizontally.

**Step 24:** Enter **-5** for the Left x.

• The value of `pointer` will be used to point at the correct picture.

• You will want to only display the picture of Ohio when the value of the variable `pointer` is equal to **1**.

**Step 25:** In the Display picture (only if ...) window, enter **pointer = 1**. (See Figure 53.)
Step 26: Click **Apply**. You may notice the picture may not appear on the Cartesian graph. The value of the variable `pointer` may not be equal to one at this point. The picture may not display until you have entered all five pictures.

- Click **OK** to record your changes and close the New Picture dialog.

Step 27: Click **OK**.

- You will want to repeat steps 19 through 27 for the other 4 states.

**Step 28:** Find and insert the bitmap of Kansas next (KSshape.bmp). Enter **Kansas** for the picture name. Resize and reposition the picture so that it fits in the preview window. In the Display picture (only if…) field, enter **pointer =2**. (See Figure 54.)

**Figure 53** New Picture - Ohio

---

**Figure 54** New Picture - Kansas
Step 29: Find and insert the bitmap of Texas next (TXshape.bmp). Enter Texas for the picture name. Resize and reposition the picture so that it fits in the preview window. In the Display picture (only if…) field, enter pointer = 3. (See Figure 55.)
Figure 55 New Picture – Texas

Step 30: Find and insert the bitmap of Michigan next (MIshape.bmp). Enter Michigan for the picture name. Resize and reposition the picture so that it fits in the preview window. In the Display picture (only if…) field, enter pointer =4. (See Figure 56.)

Figure 56 New Picture – Michigan

Step 31: Find and insert the bitmap of Florida next (FLshape.bmp). Enter Florida for the picture name. Resize and reposition the picture so that it fits in the preview window. In the Display picture (only if…) field, enter pointer =5. (See Figure 57.)
Step 32: Click Apply and OK.

• Your Functions tab should appear. (See Figure 58.)

**Figure 58** Format Graph Cartesian – Functions for all 5 states

- Most of the time when you are adding a picture or a shape to a Cartesian graph, you will want to clear the axes, background, and labels.

Step 33: With the Format Graph – Cartesian window open, switch to the Axes tab and click Clear. The axes disappear from the graph.

Step 34: Click OK.

Step 35: Click Calculate Values at least 10 times to check that the names of the states match the picture displayed.
Step 36: Click **Record** to record the question.

Step 37: Click **Save** to save the question bank.

**Use the Shape Function**

- Common shapes, such as rectangles, spheres, and triangles, are built into the software—no separate file is needed to insert these shapes into a graph.
- You can add common shapes to a Cartesian graph. Over 20 pre-defined shapes are available.
- You may insert a shape into a question, answer, rationale, feedback, matching group, or narrative.
- For the steps in this section, you will use Problem 3 to make a question containing a cylinder with changing width and height.

**Step 1:** Select **Problem 3** and click **Edit**.

**Step 2:** Click on the picture of the cylinder and press DELETE on your keyboard.

- You will now create the cylinder using the shapes provided by the software.

**Step 3:** Choose **Insert | Graph | Cartesian**.

**Step 4:** With the Functions tab displayed, click on the drop-down list and select **Shape**.

**Step 5:** Click **New**.

- Select a shape from the drop-down list or enter the name of the shape to display on the graph. The shape will be drawn in the preview window.

**Step 6:** From the Shape name drop-down list, choose **Cylinder**. A cylinder appears in the preview window. (See Figure 59.)

![Figure 59 - New Shape – Choose Cylinder](image)

- You can optionally click **Apply** to see how the shape will look on the graph in the editing dialog.

**Step 7:** Click **Apply**.

- You will now change the size and location of the cylinder.
• You will begin by creating a cylinder with static width and height.

**Step 8:** In the Height field, enter 8.

• The Top y should generally be half the height in order to center the figure vertically in the preview window.

**Step 9:** In the Top y field, enter 4.

**Step 10:** In the Width field, enter 4.

• The Left x should generally be half the width in order to center the figure horizontally in the preview window.

**Step 11:** In the Left x field, enter -2. (See Figure 60.)

Figure 60  New Shape – Cylinder

If you want the picture to display only if a certain condition is met, enter the condition in the Display picture (only if...) field. You will leave this field blank.

• You may choose a pattern, color, and shading style for the shape. You will leave all these options as is for this question.

**Step 12:** Click **Apply** to see how the shape will look on the graph in the editing dialog.

**Step 13:** Click **OK** to record your changes and close the New Shape dialog.

• You will now enter the labels for the width and height. You will use the Text Box option from the function tab drop-down list.

**Add a Text Box to the Cartesian Graph**

• You can add text boxes to a Cartesian graph as a label for pictures, as a label for axes, or simply to label portions of a graph not linked to a particular point or requiring text rotation.

• For the steps in this section, you will use add text boxes to the graph in Problem 3.

• You will begin by choosing Text Box from the drop-down list menu in the Functions tab.

**Step 14:** With the Functions tab displayed, click on the drop-down list menu and select Text Box.

**Step 15:** Click **New**.

• Enter the text to display on the graph. Click **Font** to set the font characteristics and color for the text.
**Step 16:** Enter 4 cm in the Text field. You will see the text appear in the preview window at the center of the window.

- Enter the x and y coordinate for the center of the text. The text will be drawn with half the text to the left of this point, half to the right, half above, and half below.

**Step 17:** Enter -4.5 in the Center y field. (See Figure 61.)
Step 18: Click **Apply**.

Step 19: Click **OK**.

Step 20: Confirm that **Text Box** is already selected in the drop-down list in the Functions tab and click **New**.

- Enter the text for the height.

Step 21: Enter **8 cm** in the Text field. You will see the text appear in the preview window on the center of the window.

Step 22: Enter **3** in the Center x field.

Step 23: Click **Apply** and **OK**. The text appears on the graph and the Add Text Box window closes. (See Figure 62.)

**Figure 62** Format Graph Cartesian – Static Cylinder

Step 24: Click **OK**. The Format Graph – Cartesian window closes.
• Except for clearing the axes, this problem would be complete in static form. You will now make this problem
dynamic by creating variables for the width and height of the cylinder.

**Step 25:** Choose **Edit | Algorithm Definitions.** The Edit Algorithm Definitions window opens.

**Step 26:** Click **New.**

**Step 27:** Enter **width** in the Variable name field.

• The width will be a number between 3 and 9 inclusive with increments of 1.

**Step 28:** Enter **range(3,9)** in the Definition field.

**Step 29:** Click **OK.**

• Create the height variable.

**Step 30:** Click **New.**

**Step 31:** Enter **height** in the Variable name field.

• The height will be a number between 2 and 8 inclusive with increments of 1.

**Step 32:** Enter **range(2,8)** in the Definition field.

**Step 33:** Click **OK.** (See Figure 63.)

![Figure 63](Image)

**Figure 63** Edit Algorithm Definitions – Width and height

**Step 34:** Click **Done.**

• You will now change the size and location of the cylinder using the variables for width and height.

**Step 35:** Double-click on the Cartesian graph. The **Format Graph - Cartesian** dialog box opens.

**Step 36:** With **Shape: Cylinder** selected, click **Edit.**

**Step 37:** In the Height field, enter **height.** The cylinder may temporarily disappear. When you recalculate the
question, the values for the variables will be used to provide a height and width for the question.

• The **Top y** should generally be half the height in order to center the figure vertically in the preview window.

**Step 38:** In the Top y field, enter **height/2.**

**Step 39:** In the Width field, enter **width.**
• The $Left\ x$ should generally be the opposite of half the width in order to center the figure horizontally in the preview window.

*Step 40:* In the Left x field enter $-\text{width}/2$. (See Figure 64.)
Figure 64  Edit Shape – Cylinder

**Step 41:** Click **Apply**.

**Step 42:** Click **OK** to close the Edit Shape window.

**Step 43:** Click **OK** to close the Format Graph – Cartesian window.

- Check the changes made by recalculating the question four times.

**Step 44:** Click **Calculate Values** and watch the cylinder change width and height.

- You will want to keep the axes on the graph until the all text and shapes have been correctly positioned on the graph.

- You will now make the location of the text boxes variable.

**Step 45:** Double-click on the graph. The Format Graph – Cartesian window opens.

**Step 46:** With the Functions tab displayed, select the **Text Box: 4 cm** entry and click **Edit**.

- Since the variable `width` is a number and the text “ cm” is a string, you must use the string value for `width`. You are not permitted to combine a string and a number.

- You will use the `str` function to convert the numeric value of `width` to a string value.

**Step 47:** Replace `4 cm` with `str(width) + " cm"` in the Text field.

- The placement of the width textbox is dependent on the height of the cylinder.
  - Find half the measure of the height (`height/2`).
    - Place the text below the cylinder (`-height/2`). This would place the text directly on the bottom of the cylinder.
    - Move it slightly lower by subtracting .5 from this number (`-height/2-.5`).

**Step 48:** Replace `-4.5` in the Center y field with `-height/2-.5`. (See Figure 65.)
Step 49: Click **Apply** to apply the changes.

Step 50: Click **OK** to close the Edit Text Box window.

- You will now make the height text box variable.

Step 51: With the Functions tab displayed, select the **Text Box: 8 cm** entry and click **Edit**.

- Since the variable `height` is a number and text “ cm” is a string, you must use the string value for `height`. You are not permitted to combine a string and a number.

- You will use the `str` function to convert the numeric value of `height` to a string value.

Step 52: Replace **8 cm** with `str(height) + " cm"` in the Text field.

- The placement of the height text box is dependent on the width of the cylinder.
  - Find half the measure of the width (`width/2`).
  - The text will be placed to the right of the cylinder (in a positive direction). This would place the text directly on the right side of the cylinder.
  - Move it slightly further to the right by adding 1 to this number (`((width/2)+1)`).

Step 53: Replace **3** in the Center x field with `(width/2)+1`. (See Figure 66.)
Step 54: Click **Apply** to apply the changes.

Step 55: Click **OK** to close the Edit Text Box window. (See Figure 67.)

Figure 67  Format Graph Cartesian – Edit Functions

Step 56: Click **OK** to close the Format Graph – Cartesian window.

Step 57: Click **Calculate Values** and check the placement of the width and height text boxes.

- Once the placement of the text boxes is correct, you will want to clear the axes.

Step 58: Double-click on the graph. The Format Graph – Cartesian dialog box opens.

Step 59: Choose the **Axes** tab.

Step 60: Click **Clear**.
Step 61: Click OK.

- You must create a variable *volume* for the answer. The volume of a cylinder is the *radius* (or half the width of our cylinder) squared times the height times *pi*.

Step 62: Choose Edit | Algorithm Definitions. The Edit Algorithm Definitions window opens.

Step 63: Click New.

Step 64: Enter *volume* in the Variable name field.

Step 65: Enter \((\text{width}/2)^2\)*height in the Definition field.

Step 66: Click OK.

Step 67: Click Done.

- Insert the answer into the answer field.

Step 68: Highlight 32 in the answer field.

Step 69: Choose Insert | Variable.

Step 70: Select *volume* and click Insert.

Step 71: Click Calculate Values to check your work.

Step 72: Click Record to record the question.

Step 73: Click Save to save the question bank.

- Remember to save your work frequently when building problems. If your computer malfunctions, your work would be lost if you haven’t saved often.

### USE VARIABLES IN NARRATIVES

Narratives may contain dynamic content. Any variable defined within the narrative may be used by every question attached to that narrative. A variable name used in a narrative may not be redefined in a question attached to that narrative.

- For the steps in this section, you will use Problem 4 and its narrative to show the use of variables in narratives.

Step 1: Select the narrative above Problem 4 and click Edit.

- Create variables for each of the number fields in the table.

Step 2: Choose Edit | Algorithm Definitions. The Edit Algorithm Definitions window opens.

Step 3: Click New.

- Create a variable *music* for the total number of students who preferred to play a musical instrument.

Step 4: Enter *music* in the Variable name field.

Step 5: Enter range(10,40) in the Definition field.

Step 6: Click OK.

- Create a variable *sports* for the total number of students who preferred to play a sport.

Step 7: Click New.

Step 8: Enter *sports* in the Variable name field.
**Step 9:** Enter `range(10,90)` in the Definition field.

**Step 10:** Click OK.

- Create a variable `video` for the total number of students who preferred to play video games.

**Step 11:** Click New.

**Step 12:** Enter `video` in the Variable name field.

**Step 13:** Enter `range(10,90)` in the Definition field.

**Step 14:** Click OK.

- Create a variable `bikes` for the total number of students who preferred to ride bikes or skateboards.

**Step 15:** Click New.

**Step 16:** Enter `bikes` in the Variable name field.

**Step 17:** Enter `range(10,50)` in the Definition field.

**Step 18:** Click OK.

**Step 19:** Click New.

- Create a variable `tv` for the total number of students who preferred to watch TV.

**Step 20:** Enter `tv` in the Variable name field.

**Step 21:** Enter `range(10,90)` in the Definition field.

**Step 22:** Click OK.

- Create a variable `other` for the total number of students who preferred to do something other than the choices listed.

**Step 23:** Click New.

**Step 24:** Enter `other` in the Variable name field.

**Step 25:** Enter `range(10,40)` in the Definition field.

**Step 26:** Click OK. (See Figure 68.)

**Figure 68**  Edit Algorithm Definitions – Table variables
**Step 27:** Click **Done**.

- Insert the variables into the table.

**Step 28:** Highlight the number **15** in the table.

**Step 29:** Choose **Insert | Variable**.

**Step 30:** Choose the variable **music** and click **Insert**.

**Step 31:** Repeat steps 28 through 30 to insert the variables **sports, video, bikes, tv, and other** in their respective places in the table. (See Figure 69.)

Figure 69  Insert variables in the narrative

**Step 32:** Click **Calculate Values** at least five times to check your work.

**Step 33:** Click **Record**.

**Create Bar Graphs**

- You will now create a bar graph using the software.

**Step 34:** Select Problem 4 and click **Edit**.

- Delete the image of the graph from the Answer section.

**Step 35:** Click the image of the graph in the Answer section, and press **DELETE** on your keyboard.

**Step 36:** With the cursor in the Answer section, choose **Insert | Graph | Cartesian**. The Format Graph – Cartesian dialog box opens.

- Change the background to make it lined.

**Step 37:** Choose the **View** tab.

**Step 38:** Choose **Lined** for the Background.

**Step 39:** Choose **Dark Blue** for the Line Color.

- Increase the size of the graph.
Step 40: Under Size, choose 3.0” for the Width, Height, and Baseline.

- Show only the first quadrant of the graph.

Step 41: Choose to Set graph left. Leave the x and y values at zero.

Step 42: Click Apply to apply the changes. (See Figure 70)
• Delete the x and y labels from the graph.

**Step 43:** Choose the **Axes** tab.

**Step 44:** Delete the x and y labels from the horizontal and vertical axes.

• Increment the vertical axis labels by 10 by changing the Scale.

**Step 45:** Change the Scale (units per tick) on the Vertical axis to **10**.

• Allow more room between ticks for the vertical axis by changing the ticks per inch.

**Step 46:** Change the Ticks per inch on the Vertical axis to **4**.

• You will keep the labels and tick marks on the horizontal axis until the bar graphs and labels have been properly positioned.

**Step 47:** Click **Apply** to apply the changes. (See Figure 71.)

*Figure 71*  Change settings in the Axes tab
Step 48: Click OK to close the Format Graph – Cartesian window.

- You will now create the bars for the graph. The two functions bar and bar1 allow you to create bars for the bar graph.
  - Bar contains shading on the right side of the bar; bar2 does not.

Step 49: Double-click on the graph, the Format Graph – Cartesian dialog box opens.

- Select Shape from the drop-down list.

Step 50: With the Functions tab displayed, select Shape from the drop-down list.

Step 51: Click New.

Step 52: From the Shape drop-down list, choose bar2.

- The height and Top y for each bar will be the same since you want the bottom of the bar to rest on the horizontal axis.
- The width of each bar will be 1.
- The Left x will begin at .5 and will increment by 2 for each new bar.
- The color will change from bar to bar.

- The shading will always be back slash bars as shown here:

Step 53: Type the word music in both the Top y field and the Height field.

Step 54: Enter .5 for the Left x field.

Step 55: Enter 1 in the Width field.

Step 56: Choose Blue for the Color.

Step 57: Choose the back slashes for the shading option.

Step 58: Click Apply to apply the changes. (See Figure 72.)

Figure 72 Enter settings for the music bar
**Introduction to Building Dynamic Content**

**Step 59:** Click **OK**.

- Since many of the settings are the same from bar to bar, you will copy the previous bar and edit it. You could also click **New** and input all settings in the New Shape window.

**Step 60:** With **Shape: bar 2** highlighted, press **CTRL+D** on your keyboard (**APPLE+D** on the Mac). An identical bar is created.

**Step 61:** Click **Edit** to edit this newly created bar.

**Step 62:** Enter **sports** in both the Top y and the Height fields.

**Step 63:** Change the Left x to **2.5**.

**Step 64:** Change the color to **Red**.

**Step 65:** Click **Apply** to apply the changes. (See Figure 73.)

*Figure 73* Enter settings for the sports bar

**Step 66:** Click **OK**.
**Step 67:** Repeat steps 60-66 to create the bar for students choosing Video Games. Enter 4.5 for the Left x and **Green** for the color. (See Figure 74.)

**Figure 74** Enter settings for the video games bar

![Figure 74](image)

**Step 68:** Repeat steps 60-66 for students choosing to ride bikes. Enter 6.5 for the Left x and **Dark Magenta** for the color. (See Figure 75.)

**Figure 75** Enter settings for the bikes bar

![Figure 75](image)

**Step 69:** Repeat steps 60-66 to create the bar for students choosing watching TV. Enter 8.5 for the Left x and **Dark Blue** for the color. (See Figure 76.)

**Figure 76** Enter settings for the TV bar

![Figure 76](image)
**Step 70:** Repeat steps 60-66 to create the bar for students choosing something other than the suggestions on the survey. Enter **10.5** for the Left x and **Dark Green** for the color. (See Figure 77.)
Figure 77 Enter settings for the other bar

![Image of a software interface with a bar graph and settings options.]

- You should have 6 bars and 6 functions listed in the Functions tab. (See Figure 78.)

Figure 78 Function listings for the bar graph

![Image of a software interface displaying function listings for a bar graph.]

- You should record and save your work thus far.

  **Step 71:** Click **OK** to close the Format Graph - Cartesian window.

  **Step 72:** Click **Record** to record the question.

  **Step 73:** Click **Save** ![Save icon] to save the question bank.

- You need to place the labels on the graph.

  **Step 74:** With Problem 4 selected, click **Edit**.

  **Step 75:** Double-click on the graph.
**Step 76:** With the Functions tab displayed, select **Text Box** from the drop-down list.

**Step 77:** Click **New**.

- Add the label “After School Activity” to the top of the graph.

**Step 78:** Enter **After School Activity** in the Text field.

**Step 79:** Click **Font** and choose **Bold** for the Font Style and **10** for the Size.

**Step 80:** Click **OK** to close the Font window.

- Position the label on the graph.

**Step 81:** Enter **6** in the Center x field. This centers the label right to left.

**Step 82:** Enter **100** in Center y field. This positions the label at the top of the graph.

**Step 83:** Click **Apply**.

**Step 84:** Click **OK**.

- Insert a label for the vertical axis.

**Step 85:** With the drop-down list still pointing to **Text Box**, click **New**.

**Step 86:** Type **Number of Students** in the Text field.

**Step 87:** Enter **-1.5** in the Center x field, **50** in the Center y field, and **90** in the Rotation field. (See Figure 79.)

**Figure 79** Insert vertical label

**Step 88:** Click **Apply** and **OK**.

- Insert labels for the horizontal axis. After building the first label, you will use it to duplicate and edit the remainder of the labels.

- Since each word you will be using for a label is also a variable, you will need to enclose each word for the label in quotes. Failing to do this would cause the numeric value associated with the variable to display.

**Step 89:** With the drop-down list still pointing to **Text Box**, click **New**.

**Step 90:** Enter “**Music**” in the Text field. Be sure to use quotation marks; otherwise, the numeric value for the variable *music* will display.
Step 91: Enter .5 for Center x, -7 for Center y, and 45 for Rotation. (See Figure 80.)
Step 92: Click **Apply** and **OK**.

- Duplicate the text box and change only the Center x (increment by 2 each time) and the text in the Text field.

Step 93: With Textbox: “Music” highlighted on the Functions tab, press CTRL+D on your keyboard (APPLE+D on the Mac). Another identical textbox will be created.

Step 94: Click **Edit**.

Step 95: Enter “**Sports**” in the Text field.

Step 96: Enter **2.5** in the Center x field.

Step 97: Click **Apply** and **OK**.

Step 98: Repeat steps 93 through 97 entering “**Video**” in the Text field and **4.5** for Center x.

Step 99: Repeat steps 93 through 97 entering “**Bikes**” in the Text field and **6.5** for Center x.

Step 100: Repeat steps 93 through 97 entering “**TV**” in the Text field and **8.5** for Center x.

Step 101: Repeat steps 93 through 96 entering “**Other**” in the Text field and **10.5** for Center x.

- If the labels look to be in the correct position, you will now delete the tick marks and labels from the horizontal axis.

Step 102: Choose the **Axes** tab.

Step 103: Under the Horizontal axis, deselect **Show tick marks**.

Step 104: Under the Horizontal axis, deselect **Label tick marks**.

Step 105: Click **Apply** to apply the changes. (See Figure 81.)
Step 106: Click OK to close the Format Graph – Cartesian window.

Step 107: Click Record to record the question.

Step 108: Click Save to save the question bank.

Step 109: Highlight the narrative and click Calculate Values at least 10 times to check your work.

- Note that narratives are generally connected to two or more questions.
- If you created another question attached to this same narrative, that question would also be able to access the variables found in the narrative.
- The variables found in the questions themselves will not be able to be shared with other questions in the narrative group.

CLOSE THE EXAMVIEW TEST GENERATOR

When you are finished working with the Test Generator, close the program.

- If you have a test open, save and close the test before you close the program.

- To close the program, choose File | Exit or click in the upper right corner of the window.

Step 1: Save the test.

Step 2: Choose File | Close to close the test.

Step 3: Choose File | Exit to exit the program.
CHECKLIST SUMMARY

Now that you have completed this section, are you comfortable with all the concepts presented in this section? See whether you can answer yes to each of the questions in this checklist summary. If you answer no to any question in the checklist summary, take the time to review the referenced material from this section.

☑ Can you explain the purpose of the ExamView Assessment Suite? (See What is ExamView Assessment Suite?)
☑ Can you explain the purpose of the ExamView Test Generator? (See What is the ExamView Test Generator?)
☑ Can you start ExamView? (See Start ExamView.)
☑ Can you define a variable? (See Define a Variable.)
☑ Can you format a variable? (See Format a Variable.)
☑ Can you insert a variable? (See Insert a Variable.)
☑ Can you use arithmetic operators? (See Use Arithmetic Operators.)
☑ Can you format coefficients? (See Format the Coefficients.)
☑ Can you use formatting functions? (See Use Format Functions.)
☑ Can you define conditions? (See Define Conditions.)
☑ Can you use scramble true? (See Use Scramble True.)
☑ Can you use lists? (See Use Lists.)
☑ Can you add variable points and rays to the number line? (See Use Number Line.)
☑ Can you define a function using f(x)? (See Use the Function f(x)=.)
☑ Can you use the circle function? (See Use the Circle Function.)
☑ Can you insert a picture that varies? (See Use the Picture Function.)
☑ Can you create a pre-defined shape in the software? (See Use the Shape Function.)
☑ Can you add a text box to a figure or graph? (See Add a Text Box to the Cartesian Graph.)
☑ Can you use variables in narratives? (See Use Variables in Narratives.)
☑ Can you exit the program? (See Exit the ExamView Test Generator.)
APPENDIX: SYNTAX

The ExamView Test Generator provides you with numerous pre-defined keywords and functions to use when defining variables, conditions, and user-defined functions.

- The user-defined functions will return one of three types of variables:
  - An integer variable includes positive and negative whole numbers and zero.
  - A double variable includes a double-precision floating point (i.e., the variable has a fractional part).
  - A string variable has characters or letters.

**Constants**

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>constant</td>
<td>2.71828..., used in problems involving growth or decay (or compound interest). Usually defined by the following equation: $e = \lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$</td>
</tr>
<tr>
<td>false</td>
<td>constant</td>
<td>0</td>
</tr>
<tr>
<td>inf</td>
<td>constant</td>
<td>Can be used to represent positive or negative infinity (INF).</td>
</tr>
<tr>
<td>no</td>
<td>constant</td>
<td>0</td>
</tr>
<tr>
<td>pi</td>
<td>constant</td>
<td>3.1415926.... By definition, PI is the ratio of the circumference of a circle to its diameter.</td>
</tr>
<tr>
<td>testversion</td>
<td>Pseudo constant</td>
<td>Represents the current test version (&quot;A&quot;, &quot;B&quot;, &quot;C&quot;, etc.).</td>
</tr>
<tr>
<td>true</td>
<td>constant</td>
<td>1</td>
</tr>
<tr>
<td>yes</td>
<td>constant</td>
<td>1</td>
</tr>
</tbody>
</table>

**Math Functions**

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>double</td>
<td>abs(double $x$)</td>
<td>Computes the absolute value of the argument, $x$.</td>
</tr>
<tr>
<td>ceil</td>
<td>double</td>
<td>ceil(double $x$)</td>
<td>Rounds up. The function finds the smallest integer not less than the argument, $x$.</td>
</tr>
<tr>
<td>floor</td>
<td>double</td>
<td>floor(double $x$)</td>
<td>Rounds down. The function finds the largest integer not greater than the argument, $x$.</td>
</tr>
</tbody>
</table>
### Math Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>frac</td>
<td>double</td>
<td>frac(double x)</td>
<td>Returns the fractional part of the argument, x. The return value will always be greater than or equal to 0 and less than 1.</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
<td>int(double x)</td>
<td>Returns the integer portion of the argument, x. If x is positive, it rounds down. If x is negative, it rounds up.</td>
</tr>
</tbody>
</table>

### Exponential and Logarithmic Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp</td>
<td>double</td>
<td>exp(double x)</td>
<td>Calculates the exponential ( e^x ).</td>
</tr>
<tr>
<td>ln</td>
<td>double</td>
<td>ln(double x)</td>
<td>Calculates the natural log of the argument, x. Arguments to the function must be greater than zero.</td>
</tr>
<tr>
<td>log</td>
<td>double</td>
<td>log(double x)</td>
<td>Calculates the natural log of the argument, x. Arguments to the function must be greater than zero.</td>
</tr>
<tr>
<td>log10</td>
<td>double</td>
<td>log10(double x)</td>
<td>Calculates the base 10 logarithm of the argument, x. Arguments to the function must be greater than or equal to zero.</td>
</tr>
<tr>
<td>Function</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| `logb`   | `double` | `logb(double x, double b)` calculates the base `b` logarithm of the argument, `x`. Arguments to the function must be greater than zero. Actual function is: 
\[
\frac{\log(x)}{\log(b)}
\]
| `pow`    | `double` | `pow(double x, double y)` calculates `x` to the power of `y`.                |
### Trigonometric Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>acos</td>
<td>double</td>
<td>acos(double x)</td>
<td>Computes the arc cosine of the argument, x. Arguments to the function must be in the range –1 to 1. Values are returned in the range 0 to π inclusive.</td>
</tr>
<tr>
<td>acosh</td>
<td>double</td>
<td>acosh(double x)</td>
<td>Computes the hyperbolic arc cosine of the argument, x. Arguments to the function must be greater than 1.</td>
</tr>
<tr>
<td>acot</td>
<td>double</td>
<td>acot(double x)</td>
<td>Computes the arc cotangent of the argument, x. Arguments to the function must not be equal to zero. Values are returned in the range –π/2 to π/2 non-inclusive.</td>
</tr>
<tr>
<td>acoth</td>
<td>double</td>
<td>acoth(double x)</td>
<td>Computes the hyperbolic arc cotangent of the argument, x. Arguments to the function must be less than –1 or greater than 1.</td>
</tr>
<tr>
<td>acsc</td>
<td>double</td>
<td>acsc(double x)</td>
<td>Computes the arc cosecant of the argument, x. Arguments to the function must be less than or equal to –1 or greater than or equal to 1. Values are returned in the range –π/2 to π/2 non-inclusive.</td>
</tr>
<tr>
<td>acsch</td>
<td>double</td>
<td>acsch(double x)</td>
<td>Computes the hyperbolic arc cosecant of the argument, x. Arguments to the function must not be equal to zero.</td>
</tr>
<tr>
<td>arccos</td>
<td>double</td>
<td>arccos(double x)</td>
<td>Computes the arc cosine of the argument, x. Arguments to the function must be in the range –1 to 1 inclusive. Values are returned in the range 0 to π inclusive.</td>
</tr>
<tr>
<td>arccosh</td>
<td>double</td>
<td>arccosh(double x)</td>
<td>Computes the hyperbolic arc cosine of the argument, x. Arguments to the function must be in the range –1 to 1 inclusive.</td>
</tr>
<tr>
<td>arccot</td>
<td>double</td>
<td>arccot(double x)</td>
<td>Computes the arc cotangent of the argument, x. Arguments to the function must not be equal to zero. Values are returned in the range –π/2 to π/2 non-inclusive.</td>
</tr>
<tr>
<td>arccoth</td>
<td>double</td>
<td>arccoth(double x)</td>
<td>Computes the hyperbolic arc cotangent of the argument, x. Arguments to the function must be less than –1 or greater than 1.</td>
</tr>
<tr>
<td>arccsc</td>
<td>double</td>
<td>arccsc(double x)</td>
<td>Computes the arc cosecant of the argument, x. Arguments to the function must be less than or equal to –1 or greater than or equal to 1. Values are returned in the range –π/2 to π/2 non-inclusive.</td>
</tr>
<tr>
<td>arccsch</td>
<td>double</td>
<td>arccsch(double x)</td>
<td>Computes the hyperbolic arc cosecant of the argument, x. Arguments to the function must not be equal to zero.</td>
</tr>
<tr>
<td>arcsec</td>
<td>double</td>
<td>arcsec(double x)</td>
<td>Computes the arc secant of the argument, x. Arguments to the function must be less than or equal to –1 or greater than or equal to 1. Values are returned in the range 0 to π inclusive.</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>RETURN VARIABLE TYPE</td>
<td>FORMAT</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>arcsech</td>
<td>double</td>
<td>arcsech(double x)</td>
<td>Computes the hyperbolic arc secant of the argument, x. Arguments to the function must be greater than 0 and less than or equal to 1.</td>
</tr>
<tr>
<td>arcsin</td>
<td>double</td>
<td>arcsin(double x)</td>
<td>Computes the arc sine of the argument, x. Arguments to the function must be in the range –1 to 1. Values are returned in the range $[-\pi/2, \pi/2]$ inclusive.</td>
</tr>
<tr>
<td>asinh</td>
<td>double</td>
<td>arcsinh(double x)</td>
<td>Computes the hyperbolic arc sine of the argument, x.</td>
</tr>
<tr>
<td>arctan</td>
<td>double</td>
<td>arctan(double x)</td>
<td>Computes the arc tangent of the argument, x. Values are returned in the range $[-\pi/2, \pi/2)$ non-inclusive.</td>
</tr>
<tr>
<td>arctan2</td>
<td>double</td>
<td>arctan2(double x)</td>
<td>Computes the arc tangent of $y/x$. This function produces correct results even when the resulting angle is near $-\pi/2$ or $\pi/2$. Values are returned in the range $[-\pi/2, \pi/2)$ non-inclusive.</td>
</tr>
<tr>
<td>arctanh</td>
<td>double</td>
<td>arctanh(double x)</td>
<td>Computes the hyperbolic arc tangent of the argument, x. Arguments to the function must be in the range –1 to 1, non-inclusive.</td>
</tr>
<tr>
<td>asec</td>
<td>double</td>
<td>asec(double x)</td>
<td>Computes the arc secant of the argument, x. Arguments to the function must be less than or equal to –1 or greater than or equal to 1. Values are returned in the range $[0, \pi]$ inclusive.</td>
</tr>
<tr>
<td>asech</td>
<td>double</td>
<td>asech(double x)</td>
<td>Computes the hyperbolic arc secant of the argument, x. Arguments to the function must be less than or equal to –1 or greater than or equal to 1. Values are returned in the range $[0, \pi]$ inclusive.</td>
</tr>
<tr>
<td>asin</td>
<td>double</td>
<td>asin(double x)</td>
<td>Computes the arc sine of the argument, x. Arguments to the function must be in the range –1 to 1 inclusive. Values are returned in the range $[-\pi/2, \pi/2]$ inclusive.</td>
</tr>
<tr>
<td>asinh</td>
<td>double</td>
<td>asinh(double x)</td>
<td>Computes the hyperbolic arc sine of the argument, x.</td>
</tr>
<tr>
<td>atan</td>
<td>double</td>
<td>atan(double x)</td>
<td>Computes the arc tangent of the argument, x. Values are returned in the range $[-\pi/2, \pi/2)$ non-inclusive.</td>
</tr>
<tr>
<td>atan2</td>
<td>double</td>
<td>atan2(double x)</td>
<td>Computes the arc tangent of $y/x$. It produces correct results even when the resulting angle is near $-\pi/2$ or $\pi/2$. Values are returned in the range $[-\pi/2, \pi/2)$ non-inclusive.</td>
</tr>
<tr>
<td>atanh</td>
<td>double</td>
<td>atanh(double x)</td>
<td>Computes the hyperbolic arc tangent of the argument, x. Arguments to the function must be in the range –1 to 1, non-inclusive.</td>
</tr>
<tr>
<td>cos</td>
<td>double</td>
<td>cos(double x)</td>
<td>Computes the cosine of the argument, x. The angle is specified in radians. Values are returned in the range –1 to 1 inclusive.</td>
</tr>
<tr>
<td>cosh</td>
<td>double</td>
<td>cosh(double x)</td>
<td>Computes the hyperbolic cosine of the argument, x. The actual function is: $\frac{e^x + e^{-x}}{2}$</td>
</tr>
<tr>
<td>cot</td>
<td>double</td>
<td>cot(double x)</td>
<td>Computes the cotangent of the argument, x. The angle is specified in radians. Arguments to the function must not be equal to zero.</td>
</tr>
<tr>
<td>coth</td>
<td>double</td>
<td>coth(double x)</td>
<td>Computes the hyperbolic cotangent of the argument, x. This evaluates to $\frac{\cosh(x)}{\sinh(x)}$. Arguments to the function must not be equal to zero.</td>
</tr>
</tbody>
</table>
### Mathematical Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>csc</code></td>
<td>double</td>
<td>Computes the cosecant of the argument, <code>x</code>. The angle is specified in radians. Arguments to the function must not be equal to zero.</td>
</tr>
<tr>
<td><code>csch</code></td>
<td>double</td>
<td>Computes the hyperbolic cosecant of the argument, <code>x</code>. This evaluates to <code>1 / sinh(x)</code>. Arguments to the function must not be equal to zero.</td>
</tr>
<tr>
<td><code>deg</code></td>
<td>double</td>
<td>Converts an argument, <code>x</code>, in radians to degrees.</td>
</tr>
<tr>
<td><code>rad</code></td>
<td>double</td>
<td>Converts an argument, <code>x</code>, in degrees to radians.</td>
</tr>
<tr>
<td><code>sec</code></td>
<td>double</td>
<td>Computes the secant of the argument, <code>x</code>. The angle is specified in radians.</td>
</tr>
<tr>
<td><code>sech</code></td>
<td>double</td>
<td>Computes the hyperbolic secant of the argument, <code>x</code>. This evaluates to <code>1 / cosh(x)</code>.</td>
</tr>
<tr>
<td><code>sin</code></td>
<td>double</td>
<td>Computes the sine of the argument, <code>x</code>. The angle is specified in radians. Values are returned in the range –1 to 1 inclusive.</td>
</tr>
<tr>
<td><code>sinh</code></td>
<td>double</td>
<td>Computes the hyperbolic sine of the argument, <code>x</code>. The actual function is: ( \frac{e^x - e^{-x}}{2} ).</td>
</tr>
<tr>
<td><code>tan</code></td>
<td>double</td>
<td>Computes the tangent of the argument, <code>x</code>. The angle is specified in radians.</td>
</tr>
<tr>
<td><code>tanh</code></td>
<td>double</td>
<td>Computes the hyperbolic tangent of the argument, <code>x</code>. This evaluates to <code>sinh(x) / cosh(x)</code>.</td>
</tr>
</tbody>
</table>

### String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Return Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>asc</code></td>
<td>integer</td>
<td>Computes the ASCII value of the first character in the string. If the string is empty, it returns 0.</td>
</tr>
</tbody>
</table>
| `chr`    | string      | Returns a string character with the ASCII value of the argument, `x`. If `x` is less than 32 or greater than 255, the function will return a question mark. **Note:** You should use this function to generate characters found only in the Symbol font. Once the variable is inserted into a question or narrative, you need to highlight the variable and change it to Symbol font for it to display properly.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>asc</code></td>
<td>integer</td>
<td><code>asc(string x)</code></td>
<td>Computes the ASCII value of the first character in the string. If the string is empty, it returns 0.</td>
</tr>
<tr>
<td><code>chr</code></td>
<td>string</td>
<td><code>chr(int x)</code></td>
<td>Returns a string character with the ASCII value of the argument, <code>x</code>. If <code>x</code> is less than 32 or greater than 255, the function will return a question mark. <strong>Note:</strong> You should use this function to generate characters found only in the Symbol font. Once the variable is inserted into a question or narrative, you need to highlight the variable and change it to Symbol font for it to display properly.</td>
</tr>
<tr>
<td><strong>FUNCTION</strong></td>
<td><strong>RETURN TYPE</strong></td>
<td><strong>FORMAT</strong></td>
<td><strong>DESCRIPTION</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>instr</strong></td>
<td>integer</td>
<td>instr(string x, string y)</td>
<td>Searches in the string, x, for the sub-string, y. If it finds it, the function will return the position in the string, x, where the first occurrence of y was found (between 1 and the length of string, x). If the substring is not found, the function returns FALSE (0).</td>
</tr>
<tr>
<td><strong>lcase</strong></td>
<td>string</td>
<td>lcase(string x)</td>
<td>Converts the string argument, x, to lower case. For example: lcase(&quot;ExamView&quot;) will return &quot;examview&quot;</td>
</tr>
<tr>
<td><strong>left</strong></td>
<td>string</td>
<td>left(string x, int y)</td>
<td>Returns the first y characters of the string argument, x. For example: left(&quot;ExamView&quot;, 3) will return &quot;Exa&quot;</td>
</tr>
<tr>
<td><strong>len</strong></td>
<td>integer</td>
<td>len(string x)</td>
<td>Returns the number of characters in the string argument, x, not counting the null-terminating character.</td>
</tr>
<tr>
<td><strong>ltrim</strong></td>
<td>string</td>
<td>ltrim(string x)</td>
<td>Trims spaces from the left side of the string argument, x. For example: ltrim(&quot; ExamView &quot;) will return &quot;ExamView &quot;</td>
</tr>
<tr>
<td><strong>FUNCTION</strong></td>
<td><strong>RETURN TYPE</strong></td>
<td><strong>FORMAT</strong></td>
<td><strong>DESCRIPTION</strong></td>
</tr>
<tr>
<td><strong>mid</strong></td>
<td>string</td>
<td>mid(string x, int start, int y)</td>
<td>Returns y characters starting at the start position in the string argument, x. For example: mid(&quot;ExamView&quot;, 2, 4) will return &quot;xamV&quot;</td>
</tr>
<tr>
<td><strong>right</strong></td>
<td>string</td>
<td>right(string x, int y)</td>
<td>Returns the last y characters of the string argument, x. For example: right(&quot;ExamView&quot;, 3) will return &quot;iew&quot;</td>
</tr>
<tr>
<td><strong>rtrim</strong></td>
<td>string</td>
<td>rtrim(string x)</td>
<td>Trims spaces from the right side of the string argument, x. For example: rtrim(&quot; ExamView &quot;) will return &quot; ExamView&quot;</td>
</tr>
<tr>
<td><strong>sentence</strong></td>
<td>string</td>
<td>Sentence(string x)</td>
<td>Changes string argument, x, into a sentence that can be subdivided using the choose function. This allows us to create a sentence that can word wrap. For example: mySentence = sentence(&quot;This is along sentence that might need to wrap.&quot;) partial choose(3, mySentence) to access the 3rd tenth of the variable mySentence.</td>
</tr>
<tr>
<td><strong>str</strong></td>
<td>string</td>
<td>str(double x)</td>
<td>Returns a string representation of the argument, x. For example: str(18) will return &quot;18&quot;</td>
</tr>
<tr>
<td><strong>strdup</strong></td>
<td>string</td>
<td>strdup(string x, int numTimes)</td>
<td>Returns a string by combining the string argument, x, a variable number of times. For example: strdup(&quot;(2)&quot;, 5) will return &quot;(2)(2)(2)(2)(2)&quot;</td>
</tr>
<tr>
<td><strong>trim</strong></td>
<td>string</td>
<td>trim(string x)</td>
<td>Trims spaces from both the left and the right side of the string argument, x. For example: trim(“ ExamView “) will return “ExamView”</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ucase</strong></td>
<td>string</td>
<td>ucase(string x)</td>
<td>Converts the string argument, x, to upper case. For example: ucase(&quot;ExamView&quot;) will return &quot;EXAMVIEW&quot;</td>
</tr>
<tr>
<td><strong>val</strong></td>
<td>double</td>
<td>val(string x)</td>
<td>Returns a numeric representation of the string argument, x. For example: val(&quot;18&quot;) will return 18 val(&quot;-4&quot;) will return -4</td>
</tr>
</tbody>
</table>
## Statistical Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>comb</strong></td>
<td>integer</td>
<td>comb(int m, int n)</td>
<td>Returns the number of possible combinations. If we have a group of ( m ) objects, how many ways can they be grouped as ( n ) objects when position does not matter? The actual function is: ( \text{comb}(m, n) = \frac{m!}{(n!)(m-n)!} )</td>
</tr>
<tr>
<td><strong>makelist</strong></td>
<td>list</td>
<td>makelist(int numValues, double lowerExtreme, double lowerQuartile, double median, double upperQuartile, double upperExtreme, Boolean sortData)</td>
<td>Use this function to create a &quot;list&quot; container to hold a group of data with ( numValues ) data, and the range, median, and interquartile ranges assigned. Pass ( \text{TRUE} ) if you want the data elements to be arranged in ascending order. Pass (-1) for either extreme, either quartile, or the median if you do not require them to be specific values. For example: ( \text{makelist}(3, 5, -1, 8, -1, 9, \text{TRUE}) ) will return &quot;5, 8, 9&quot; Both quartile values must be supplied to the makelist() function. If either one is not, they are both ignored.</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td>double</td>
<td>mean(list data)</td>
<td>Returns the arithmetic mean (or average) of the data in the list.</td>
</tr>
<tr>
<td><strong>median</strong></td>
<td>double</td>
<td>median(list data)</td>
<td>Returns the middle value of the data in the list if the number of elements in the list is odd, or the average of the two middle terms if the number of elements is even.</td>
</tr>
<tr>
<td><strong>mode</strong></td>
<td>double</td>
<td>mode(list data)</td>
<td>Returns the most common value for the data in the list. If the data is multimodal, this function returns zero.</td>
</tr>
<tr>
<td><strong>perm</strong></td>
<td>integer</td>
<td>perm(int m, int n)</td>
<td>Returns the number of possible permutations. Given that position is important, if one has ( m ) objects, how many unique ways can they be placed in ( n ) positions. The actual function is: ( \text{perm}(m, n) = \frac{m!}{(m-n)!} )</td>
</tr>
<tr>
<td><strong>sdev</strong></td>
<td>double</td>
<td>sdev(list data)</td>
<td>Returns the standard deviation of the data values in the list. Data values are assumed to be the entire population (not a sample). Result is the square root of the population variance. (see ( \text{variance} )).</td>
</tr>
<tr>
<td><strong>sort</strong></td>
<td>list</td>
<td>sort(Boolean sortAscending, double arg1, double arg2, ...) - or - sort(Boolean sortAscending, list dataList)</td>
<td>This function will sort the list of arguments. All of the arguments must be numeric (or the list must have been composed of only numeric data). Function will return a list that can be used as an argument for choose(). ( \text{sort}(&lt;\text{TRUE}&gt;, 1, 14, 3.2, 6, 9) ) will return ( 1, 3.2, 6, 9, 14 ) - or - ( \text{listOfData} = \text{list}(12, -3, 5) ) ( \text{sort}(&lt;\text{FALSE}, \text{listOfData}) ) will return &quot;12, 5, -3&quot;</td>
</tr>
</tbody>
</table>
## Logical Operators

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>expression AND expression</td>
<td>Returns TRUE (1) only if both expressions evaluate to non-zero, otherwise returns FALSE (0). If the first expression evaluates to FALSE, the second expression is not evaluated.</td>
</tr>
<tr>
<td>eqv</td>
<td>expression EQV expression</td>
<td>Returns TRUE (1) if both expressions evaluate to non-zero or if both expressions evaluate to zero, otherwise returns FALSE (0).</td>
</tr>
<tr>
<td>imp</td>
<td>expression IMP expression</td>
<td>Returns TRUE (1) if the first expression evaluates to non-zero and the second expression evaluates to zero, otherwise returns FALSE (0).</td>
</tr>
<tr>
<td>not</td>
<td>NOT expression</td>
<td>If expression evaluates to non-zero, this will return FALSE (0), otherwise it will return TRUE (1).</td>
</tr>
<tr>
<td>or</td>
<td>expression OR expression</td>
<td>Returns TRUE (1) if either expression evaluates to non-zero, otherwise returns FALSE (0). If the first expression evaluates to TRUE, the second expression is not evaluated.</td>
</tr>
<tr>
<td>xor</td>
<td>expression XOR expression</td>
<td>Returns the bitwise exclusive-OR or the two expressions.</td>
</tr>
</tbody>
</table>

## Operators

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod</td>
<td>expression MOD expression</td>
<td>Divides the first expression by the second expression and returns the remainder. If the second expression is equal to zero, the operation will return zero. The result of this operation is always non-negative.</td>
</tr>
</tbody>
</table>
# Fractions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| fracs*    | string                | fracs(double numer, double denom) | After reducing the fraction $\frac{\text{numer}}{\text{denom}}$, this function will return a string as either a whole number, or a fraction in the form "numer/denom". When displayed, this function will draw a stacked fraction if necessary. If $\text{denom}$ is 0, the function will return "0". For example:  
  \[
  \text{fracs}(22, 6) \quad \text{will return } \frac{11}{3}
  \]
  \[
  \text{fracs}(22, 11) \quad \text{will return } 2
  \] |
| mixfracs* | string                | mixfracs(double numer, double denom) | After reducing the fraction $\frac{\text{numer}}{\text{denom}}$, this function will return a string as either a whole number, or a fraction in the form "whole/numer/denom". When displayed, this function will draw a stacked fraction if necessary. If $\text{denom}$ is 0, the function will return "0". For example:  
  \[
  \text{mixfracs}(22, 6) \quad \text{will return } \frac{3}{3}
  \]
  \[
  \text{mixfracs}(2, 6) \quad \text{will return } \frac{1}{3}
  \] |
| rfracs*   | string                | rfracs(double numer, double denom) | Reduces the fraction $\frac{\text{numer}}{\text{denom}}$. When displayed, this function will always draw a stacked fraction. If $\text{denom}$ is 0, the function will return "0". For example:  
  \[
  \text{rfracs}(22, 6) \quad \text{will return } \frac{11}{3}
  \]
  \[
  \text{rfracs}(22, 11) \quad \text{will return } \frac{2}{1}
  \] |
| sfracs*   | string                | sfracs(double numer, double denom) | After reducing the fraction $\frac{\text{numer}}{\text{denom}}$, this function will return a string as either a whole number, or a fraction in the form "numer/denom". When displayed, this function will draw a small stacked fraction if necessary. If $\text{denom}$ is 0, the function will return "0". For example:  
  \[
  \text{sfracs}(22, 6) \quad \text{will return } \frac{11}{3}
  \]
  \[
  \text{sfracs}(22, 11) \quad \text{will return } 2
  \] |
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| smixfracs* | string               | smixfracs(double numer, double denom) | After reducing the fraction `numer / denom`, this function will return a string as either a whole number, or a fraction in the form "whole/numer/denom". When displayed, this function will draw a small stacked fraction if necessary. If `denom` is 0, the function will return "0". For example:  

\[
\text{smixfracs(22, 6)} \quad \text{will return} \quad 3 \frac{2}{3}
\]

\[
\text{smixfracs(2, 6)} \quad \text{will return} \quad 1 \frac{1}{3}
\] |
| srfracs* | string               | srfracs(double numer, double denom) | Reduces the fraction `numer / denom`. When displayed, this function will always draw a small stacked fraction. If `denom` is 0, the function will return "0". For example:  

\[
\text{srfracs(22, 6)} \quad \text{will return} \quad 1 \frac{11}{3}
\]

\[
\text{srfracs(22, 11)} \quad \text{will return} \quad 2 \frac{2}{1}
\] |
## Special Functions

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>choose</td>
<td>choose(int which, expression arg1, expression arg2, ...)</td>
<td>This function will select the argument at the which location in the list of arguments. The expression will be evaluated and returned. All of the arguments must be of the same type (string or numeric). If which is greater than the number of arguments, the function will use the remainder part of which to choose an argument. For example: which = rand(5) choose(which, 1, 14, 3.2, 6, 9) will return 3.2 when which = 3 - or - which = rand(5) listData = list(1, 14, 3.2, 6, 9) choose(which, listData) will return 9 when which = 5</td>
<td></td>
</tr>
<tr>
<td>decs*</td>
<td>string decs(double x)</td>
<td>This function returns a string representation of the decimal number. If the function determines that the number, x, has an infinitely repeating decimal part, the repeating part will be displayed with an overbar. For example: decs(18.333333333333) will return 18.3 decs(0.0142857142857) will return 0.0142857142857 decs(4.25) will return 4.25</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>string format(string format, double x)</td>
<td>Writes formatted output to a string. For a complete description of formatting specifiers, look in any C programming manual. For example: format(&quot;$%1.2f&quot;, 13.4) will return &quot;$13.40&quot; format(&quot;%d feet&quot;, 17) will return &quot;17 feet&quot;</td>
<td></td>
</tr>
<tr>
<td>gcf</td>
<td>integer gcf(int x, int y)</td>
<td>Returns the greatest common factor of the arguments, x and y</td>
<td></td>
</tr>
<tr>
<td>if</td>
<td>expression if (int condition, expression arg1, expression arg2)</td>
<td>Use to implement a conditional expression. If condition evaluates to non-zero, arg1 will be returned, else arg2 will be returned. Both of the arguments must be of the same type (string or numeric)</td>
<td></td>
</tr>
<tr>
<td>isalpha</td>
<td>integer isalpha(string x)</td>
<td>If the first character of the string argument, x, is in the range &quot;A&quot; to &quot;Z&quot; or &quot;a&quot; to &quot;z&quot;, the function will return TRUE (1), otherwise it will return FALSE (0)</td>
<td></td>
</tr>
<tr>
<td>isdigit</td>
<td>integer isdigit(string x)</td>
<td>If the first character of the string argument, x, is in the range &quot;0&quot; to &quot;9&quot;, the function will return TRUE (1), otherwise it will return FALSE (0)</td>
<td></td>
</tr>
<tr>
<td>FUNCTION</td>
<td>RETURN VARIABLE TYPE</td>
<td>FORMAT</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>isprime</td>
<td>integer</td>
<td>isize(int (x))</td>
<td>Evaluates the number to see if it is prime. If the number is prime, the function will return TRUE (1), otherwise it returns FALSE (0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><em>Note:</em> Values must be in the range 2 to 1,000,000.</td>
</tr>
<tr>
<td>isrelprime</td>
<td>integer</td>
<td>isize(int (x), int (y))</td>
<td>Evaluates the two numbers to see if they are relatively prime. If the greatest common factor of (x) and (y) is 1, the function will return TRUE (1), otherwise it returns FALSE (0).</td>
</tr>
<tr>
<td>isunique</td>
<td>integer</td>
<td>isize(expression (arg1), expression (arg2), ...)</td>
<td>Compares the arguments to see if there are any duplicates. If there are duplicate arguments, the function will return FALSE (0), otherwise it returns TRUE (1). All of the arguments must be of the same type (string or numeric). This function is useful as a &quot;condition&quot; in evaluating multiple choice answer choices. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(condition) isunique(answerA, answerB, answerC, answerD)</td>
</tr>
<tr>
<td>lcd</td>
<td>integer</td>
<td>isize(int (x), int (y))</td>
<td>Returns the least common denominator of the arguments, (x) and (y).</td>
</tr>
<tr>
<td>list</td>
<td>list</td>
<td>isize(expression (arg1), expression (arg2), ...)</td>
<td>Use this function as a container to hold a group of similar expressions. The expressions must all be of the same type either numeric or string. If you create a list of strings, you must make sure that no string element contains a comma, since the comma is used to delimit the list. A variable defined as a &quot;list&quot; can be used as an argument for the &quot;choose&quot; function. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fruitList = list(&quot;apple&quot;, &quot;pear&quot;, &quot;peach&quot;, &quot;tangerine&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>whichFruit = rand(4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fruit = choose(whichFruit, fruitList)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>weightList = list(110, 142, 153, 180, 212)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>whichWeight = rand(5)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>weight = choose(whichWeight, weightList)</td>
</tr>
<tr>
<td>prime</td>
<td>integer</td>
<td>isize(int (x), int (y))</td>
<td>Returns a pseudo-random prime number in the range (x) to (y), inclusive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><em>Note:</em> Values must be in the range 2 to 1,000.</td>
</tr>
<tr>
<td>range</td>
<td>double</td>
<td>isize(double (x), double (y), double increment (optional))</td>
<td>Returns a pseudo-random number in the range (x) to (y), inclusive. If the optional argument, <code>increment</code>, is included, the function will only select values in the given range that are offset from the lower limit by <code>increment</code>.</td>
</tr>
</tbody>
</table>
|              |                      |                         | *Note:* If `increment` is in the form \(.1\), \(.01\), \(.001\), etc., then the result will always have the same number of decimal places as the `increment`. Also, if \(x\) is less than zero, and \(y\) is greater than zero, and an `increment` is provided, then the value 0 will never be returned by the function.
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| **round** | double | `round(double x, int precision)` | Rounds the argument, `x`, to have `precision` decimal places. **Note:** If `precision` is greater than or equal to 10, this function rounds the argument, `x`, to the nearest `precision`. For example: 
  - `round(3.14159265, 4)` will return 3.1416 
  - `round(1492.14, 10)` will return 1490 
  - `round(1492.14, 100)` will return 1500 |
| **sciens** | string | `sciens(double x)` | Function returns a string representation of the argument, `x`, in scientific notation. **Note:** You must be sure to also change the format to Scientific Notation and select the number of decimal places when formatting the variable. For example: 
  - `sciens(143948.123)` will return `1.44 \times 10^5` when using 2 decimal places. |
| **sgns** | string | `sgns(double x)` | If the argument, `x`, is less than zero, this function returns `"-"`. Otherwise it returns `"+"`. |
| **sigfig** | string | `sigfig(double x, int numSigFigs, Boolean forceNumSigFigs)` | Creates a string version of the argument, `x`, that has `numSigFigs` significant digits. If `forceNumSigFigs` is TRUE, then the string returned will have exactly `numSigFigs` significant digits (even if the function has to randomly create some digits). If `forceNumSigFigs` is FALSE, then the function will return a string that has `numSigFigs` significant digits or less. For example: 
  - `sigfig(12362, 3, FALSE)` will return 12400 
  - `sigfig(2.14, 5, TRUE)` will return 2.1400 
  - `sigfig(1400, 4, TRUE)` might return 1427 |
| **sqrs** | string | `sqrs(double x)` | After reducing the argument, `x`, to a multiple of a whole number and a radical, it returns a string representation in the form whole/radical. When displayed, this function will draw a radical if necessary. For example: 
  - `sqrs(18)` will return `3\sqrt{2}` 
  - `sqrs(16)` will return 4 
  - `sqrs(-8)` will return `2i\sqrt{2}` 
  - `sqrs(3.14159)` will return `\sqrt{3.14159}` |
<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>RETURN VARIABLE TYPE</th>
<th>FORMAT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| symbol*    | string               | symbol(string x) | Converts the string argument, x, to the appropriate Symbol character. **Note:** Once the variable is inserted into a question or narrative, you need to highlight the variable and change it to Symbol font for it to display properly. The following list shows valid arguments to this function. For all other Symbol characters, use the chr function.  
  symbol ("\(\leq\)) will return ≥  
  symbol ("\(\geq\)) will return ≤  
  symbol ("\(<\)\(\neq\)) will return ≠  
  symbol ("\(\times\)) will return ×  
  symbol ("\(\cdot\)) will return •  
  symbol ("\(\div\)) will return ÷ |
| unsort     | list                 | unsort(list dataList) | This function will "unsort" or scramble the list of arguments. The list passed in as an argument must have been composed of only numeric data. Function will return a list that can be used as an argument for choose( )  
  listofData = list (1, 2, 3, 4)  
  unsort(ListofData) might return "2, 4, 3, 1" |

* Keywords marked with an asterisk are for display only and cannot be used as part of an expression.