# Table of Contents

Table of Contents .................................................................................................................................................... 1

Index of Command Usage........................................................................................................................................ 2

1 Introduction and a Brief History of EDU/MapleTA ............................................................................................. 3

2 Implementing Algorithms in MapleTA .................................................................................................................. 7

3 Randomized Multiple Choice ............................................................................................................................. 10

4 Randomized Text Substitution ............................................................................................................................ 15

5 Randomized Images in Multiple Choice ............................................................................................................. 18

6 Nested Statements and Conditionals .................................................................................................................. 22

7 Random Multiple Selection ................................................................................................................................ 28

8 Introduction to Numeric Questions ..................................................................................................................... 36

9 Limiting Randomization with Condition Statements .......................................................................................... 41

10 Overview of Question Designer ........................................................................................................................ 46

11 Narratives in the Question Designer ................................................................................................................ 51

12 Creating Questions using Tables with Randomized Entries ........................................................................... 57

13 Randomized Table Configurations .................................................................................................................... 63

14 Case Study – Randomized Planetary Positions ............................................................................................... 68
## Index of Command Usage

<table>
<thead>
<tr>
<th>Command</th>
<th>1 History</th>
<th>2 Algorithms</th>
<th>3 Multiple Choice</th>
<th>4 Text</th>
<th>5 Images</th>
<th>6 Conditionals</th>
<th>7 Multiple Selection</th>
<th>8 Calculations</th>
<th>9 Limiting</th>
<th>10 Question Designer</th>
<th>11 Feedback</th>
<th>12 Narration</th>
<th>13 HTML</th>
<th>14 Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>rint()</td>
<td>x x x x x x</td>
<td>x</td>
<td>x x</td>
<td>x x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>switch()</td>
<td>x x x x x</td>
<td>x</td>
<td>x x</td>
<td>x x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>strcat()</td>
<td>x x</td>
<td></td>
<td>x x</td>
<td>x x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if()</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>equalities</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>condition:</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>range()</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sig()</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>decimal()</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rand()</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abs()</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>math</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>numfmt()</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$()</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction and a Brief History of EDU/MapleTA

This initial chapter will serve as an introduction to creating randomized content in MapleTA. It will summarize the advantages to having such content, the history of such capabilities as MapleTA has evolved, idiosyncrasies that affect such content (which may not mean much to you until you get further in the manual), the organization of this manual, and the intended audience.

The Benefits of Randomization

There are tremendous advantages to using randomized content in web-based assessment:

- The technique allows each student in a class to receive a unique version of an on every attempt.
- It discourages memorization and unproductive communication between students (“the answer to question one is 14.2º N”) and encourages communication focusing on ideas and techniques (“The test really emphasizes applications of Newton’s second law with uniform circular motion. Make sure you know that well”).
- It allows students to practice and receive feedback.

There are two distinct approaches to creating randomized content: 1) randomly selecting questions from pools of comparable questions and 2) randomizing individual questions through adding additional commands. This manual will focus entirely on the second method as the first depends strongly upon an instructor’s existing content. Creating effective randomized web-based assessments typically involves some combination of the two approaches.

The History of Randomized Questions in EDU/MapleTA

The web-based assessment engine we know today as MapleTA has a long history at UNL having first been created in 1996 (as eGrade) by former Math Professor John Orr. UNL partnered with Brownstone Learning starting ~2000 to help develop the product to its complete potential and disseminate the product now known as EDU nationally. Prior to ~2004 the only method for creating questions was editing .qu codes in a text editor offline and then uploading them.

This approach has always been burdened by the need to keep track of numbering (and renumbering after inserting a question in a testbank) and poor documentation of the coding commands. These obstacles created a
steep learning curve encouraging most users to keep it simple. When one would upload .qu codes they were given either a “success” message or a fairly clear list of errors with line numbers where the errors occurred. An iterative method of fix an error and retry ultimately led to success for persistent users.

Early versions of an online GUI editor (~2005) were buggy with limited functionality and most faculty who were familiar with .qu codes stuck with that method. However, even at this early state the online editor was useful for quickly creating the framework to a question.

Around this time UNL increased its number of partners to include MapleSoft, a company well-known for the symbolic algebra program Maple. It was hoped that they would advance the mathematical capabilities of EDU. In 2007 there was a falling out between Brownstone Learning and MapleSoft and both companies went in separate directions while each retained a copy of the EDU Code. MapleSoft continued development of EDU calling it MapleTA while Brownstone Learning (later purchased by Horizon Wimba) did not. More recently (~2008) usage of EDU was discontinued on the UNL campus. We have since focused on MapleTA (still actively being developed by MapleSoft) and most recently exclusively MapleTA running as a plugin of Blackboard.

Over the past 5 years, there has been a slow, steady movement away from .qu codes while MapleSoft encouraged use of the online editor:

- The uploading of .qu codes was eliminated in MapleTA (~2009) and using the online editor was the only real method for creating questions.
- The capability to upload .qu codes was later restored (~2010). However, a unique identifier (the uid field) needed to be defined in each question to prevent the system from being confused about which version of which question was being used. Thus, users were required to upload .qu codes which created the uid field in the code and then copy the uid field from the online editor back into the .qu codes for future use.
- The MapleTA system no longer provided any debugging information when .qu codes are uploaded (~2011). This effectively puts usage of .qu codes at such a tremendous disadvantage that it is very difficult to use even for experienced .qu coders. You really need to place only one complex question in a file and finish that question before returning it to a question bank file with other questions. (Some of that debugging information has recently been restored.) It still might be advantageous to use .qu codes when moving a bunch of simple questions from another format into MapleTA. After getting the questions into the system, additional work can be done with the formatting in the online editor.

There is presently such limited functionality for uploading .qu codes into MapleTA that we no longer recommend this method for question creation.

The online editor has continued to improve over time. The number of features and ease of use have increased, while crashes now appear to be a thing of the past. There have been a number of significant developments very recently:

- It is now possible to place a variable in a question’s answer field using the online editor (as was always possible in .qu codes). This allows simple question types like multiple choice and multiple selection to have a large number of randomized permutations with a small amount of code.
-Resizable windows allow much more complex code to be created easily as you can now enlarge the window and “see it all”.
• It is now much easier to share content. In addition to teaching users how to create randomization from scratch, this manual will illustrate how to take an existing randomized question (shared through the MapleTA cloud) and edit it in the user’s content area to use it as the basis for a new randomized question.

This manual will focus almost exclusively on question creation in the online GUI editor.

Questions Types and the Question Designer

This manual will cover a large number of MapleTA question types: True/False, Multiple Choice, Multiple Selection, Numeric, and others. Code to govern randomization can be added to all of these question types. In addition, there is a sophisticated question type known as a Question Designer question (formerly called an Inline question) which is created in the MapleTA question designer. It is the evolution of earlier multipart questions. Question Designer questions allow you to connect several other questions of different type together and create uniquely structured combinations – they can be as sophisticated as web pages with areas for students to enter gradable responses. This has powerful possibilities for creating tutorials as students are asked to actively participate in the flow of information and quickly receive a score and feedback on their ability to do so.

However, this created two possible paths for the authors of this document. Should we teach users to 1) create Multiple Choice questions with randomization code in them or 2) create questions in the Question Designer with a single Multiple Choice component. They are effectively identical, but require slightly different training to create. We believe that a good fraction of the users of this manual will be faculty who have previously used MapleTA and have existing Multiple Choice questions to which they wish to add randomization code which suggests creating randomization by Method #1. Yet the question designer (Method #2) allows one to create much more sophisticated assessments, although the many more features that can be a little intimidating at times.

We have included both approaches in this manual. Method #1 is what most users will be interested in at the start (early in the manual). However, Method #2 is where you ultimately want to end up (later in the manual).

An even larger issue is how should Maple code be handled. This manual is very heavily based upon commands that were in the old versions of EDU. Although all of this code runs in MapleTA, a much larger more sophisticated programming language based upon Maple also runs in MapleTA. As an example of this, note that there were no arrays (a commonly used programming structure) in EDU and you will see that our code examples in this manual often go to great lengths to avoid arrays. Yet arrays are commonly used in MapleTA (in code examples from other universities) as they have been a major part of Maple programming for years. Faculty at UNL have a unique perspective on MapleTA due to UNL’s EDU background. Most other users worldwide see things differently as they come from a Maple background (and the vast majority of them are mathematicians). We will gradually include Maple code in this manual as we become more familiar with it ourselves.
Organization of this Manual

This manual has been created in a very modular fashion. Each section illustrates and explains a chunk of MapleTA code with unique capabilities. The sections have been named descriptively in the Table of Contents and begin with a short description of their contents so that users can effectively hop around. We have proceeded under the assumption that very few disciplines are likely to be interested in the entire manual and hope users can find those sections of interest to them. We have also created an index of MapleTA programming commands. This will enable users to move to a section defining a command and illustrating its usage after seeing it for the first time.

Intended Audience

This document has multiple intended audiences. We hope that it makes randomization accessible to new MapleTA users and encourages them to adopt best practices related to both randomization and providing feedback. We also think that it will help experienced MapleTA users add randomization to existing questions. Lastly, we hope that this document is useful for experienced .qu coders. Most of these faculty will know randomization, but could use help in moving to effective usage of the online editor. They are also unlikely to know all of the techniques displayed in this manual.

Note that another way of viewing UNL’s unique background is noting the large number of non-mathematical disciplines where EDU was used – and very effectively used due to its flexibility. The mathematical randomization capabilities of MapleTA are extensive and well-documented. This manual will focus far more on the randomization of text (which is not well-covered in MapleTA documentation) and is primarily aimed at non-mathematical disciplines and instructors of descriptive science courses.

This manual is likely to be a work in progress for some time. Please don’t hesitate to share your thoughts with the MapleTA Support Team.
Implementing Algorithms in MapleTA

MapleTA is a powerful tool for creating questions with specific answers, but we want to expand on this capability and create questions that have randomized components. This can be as simple as presenting a true or false statement in different forms (e.g. “Is 5 greater than 3?” or “Is 5 less than 3?”) or as sophisticated as an accounting ledger problem dealing with randomly generated quantities and costs. We can also create complicated permutations of problems to compare items based on numeric criteria that we provide.

Randomizing questions allows for their reusability by students. Since the problems will change each time students see them, they have no incentive to memorize the answers to problems. Creatively mixing components of a question forces the student to truly engage with the question rather than just identify the problem and answer based on familiarity.

To accomplish this task of randomization, we will use variables defined in the Algorithm Field when making our question. This and subsequent tutorials are practical explanations of some of the content available on algorithms in the MapleTA Users Guide v9.0 (Section 10.1).

Variables in MapleTA

Variables in MapleTA can be used in any of the fields (Question Designer Form, Feedback, Algorithm, Information Fields, Hints, and Solution). MapleTA variables…

- always start with a dollar-sign ($).
- are case sensitive.
- must start with a letter.
- can only contain numbers and letters (as an example, underscores are not allowed).
- are defined in the algorithm field.
- are untyped. (MapleTA treats everything as a string unless it has a reason to do otherwise)

There are no reserved variables (when defined in the algorithm field). So there no disallowed variable names as long as you follow the requirements. Manipulation of variables only happens in the Algorithm field (with one exception covered in Chapter 10). There is a GUI method for manipulating variables using Algorithms, but it is cumbersome. Therefore, this manual focuses on modifying algorithm code to manipulate variables.

The equals sign is used to declare of variables. Many other conventions that are common in other programming languages hold true here as well: arithmetic can be done on numbers, strings require encapsulation in double-quotes, and variables can be passed into functions. Every variable declaration was required to end with a semicolon in .qu codes as an end-of-statement character, but this is optional in the GUI editor. One thing that is different than most other languages is that variables cannot be defined in terms of themselves – so there is no simple increment and decrement operation.

Assignment of a variable in a line of MapleTA code is done with the equal sign and has the value on the right and the variable name on the left. Some examples of variable declarations follow:
The variables are now set to: $num = 3, $num1 = 3, $num2 = 8, $text = “This is text.”, and $text1 = “This is text with the number $num1.”. Any of these can be referenced in the Question Designer Form, Feedback or any other field which sets the output. Note that these variable values will be the same each time that this block of code executes in MapleTA.

The Algorithm Field

Within the algorithm field we can use randomization, logic, and simple control structures to modify variables. To create a randomized question we will be generating random numbers and these numbers will be used to determine what output text we use.

The simplest example of this is with a true or false question. Suppose we have the question and answer (with possible choices given in brackets and separated by a forward slash): Q: “Sir Isaac Newton is responsible for the \[Law of Universal Gravitation / Theory of Relativity\]?” A: [True / False]. In one situation our question is going to ask about the Law of Universal Gravitation, and in another situation our question is going to ask about the Theory of Relativity.

We can also define the answer differently based on the replaced component in the question. A single randomized number with a 50/50 result can help to produce two questions regarding the same general body of knowledge. We could do this with three variables: a random number, our text for the question related to the random number, and the answer related to the same random number (so the problem and answer are aligned).

Written in MapleTA’s Question Designer Form, our question would look something like “Sir Isaac Newton is responsible for the $problem?” where we have defined $problem to be defined as one of our choices: “Law of Universal Gravitation” or “Theory of Relativity”. The variable defining our answer would also be set accordingly (these specifics are in the next chapter). These variable values will NOT be the same each time that this block of code executes in MapleTA.

Question Flow

One of the critical things to understand with using variables and the algorithm field is that all variables are determined and set once when the question is requested. The variable declarations in the algorithm field are determined, the output fields are decided upon, and then the question is displayed to the student. No additional ‘calculation’ is done within the algorithm field after the student first sees a question.
With this in mind, it is important that all of the fields visible to the student have the appropriate feedback. For some questions, feedback can be made static. However, for some others feedback must be dynamic to make sense in terms of the random components of the question.

The values of generated randomized variables are stored in MapleTA and retrieved for all future displays of that instantiation of the question.
3 Randomized Multiple Choice

We also have the ability to create a multiple choice question that presents itself in a different form each time a student loads the question. This is especially useful because it allows one question to represent all aspects of a concept through reuse rather than a single aspect of that concept. The simple concepts contained in this chapter are likely to be the most widely used MapleTA capabilities covered in this entire manual.

After Completing This Tutorial You Should...

- Be able to use rint() and switch() in the Algorithm Field
- Be able to create a multiple choice question with a randomized answer

New Functions

These commands will get entered into the Algorithm Field and will affect what appears in the Question Design Form. We will make use of two different MapleTA programming commands in this tutorial:

rint(int n) - randomly selects an integer between 0 and (n-1)

switch(int n, arg0, arg1, ..., argn) - selects argn from the argument list

Example 3.1 – Creating a Multiple Choice Question With Randomization

Begin by selecting Questions - New Question. Then select to be Multiple Choice as the Question Type and enter a Question Description of Randomized MC on Lunar Phases. Our question will simply ask students what time a given lunar phase rises but will randomly choose between two different phases.

The code in Figure 3-1 is explained below:

Line 1: Assigns either a 0 or 1 to $num1, each 50% of the time

Line 2: Assigns either "waxing gibbous" or "waning gibbous" to $moonPhase

Line 3: Assigns the correct answer to $rightPhase

Line 4: Assigns the wrong answer to $wrongPhase

Note that clicking the Refresh button (in the form, not your browser!) repeatedly is useful for getting a feeling for the possible values of randomization and catching errors.

Using Variables

Earlier versions of MapleTA required you to define variables in the Algorithm field before they could be saved in questions. This is an evolving area! You can now save undefined variables in a question, but they appear in a “confusing” form until defined.

It may be best to initially create a dummy variable like (moonPhase = “test”) in the algorithm field while making your question. It is simply mentally difficult to create all of the variables that you need in a question before it is written, but it may be quickest to do a little planning on scratch paper before beginning a question so that this is possible. You MUST first define any variables that you place in the answer field or you cannot save the question – a dummy variable here like ($answer = 1;) is a necessity.
Figure 3-1 – Randomized Multiple Choice Algorithm Field

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>num1</td>
<td>1</td>
</tr>
<tr>
<td>moonPhase</td>
<td>waning gibbous</td>
</tr>
<tr>
<td>rightPhase</td>
<td>9 pm</td>
</tr>
<tr>
<td>wrongPhase</td>
<td>3 pm</td>
</tr>
</tbody>
</table>

Figure 3-2 – A Randomized Multiple Choice Question Text shown in Source Mode

The $moonPhase moon rises at ...

Choices for the answers:
(Click on the box beside an answer to mark it as correct)
- $rightPhase
- $wrongPhase
- 6 pm
- midnight
- 6 am

Change the order of the choices?
- Yes
- No

Allow more than one selection?
- Yes
- No
Note that the correct answer is always in the same slot of a multiple choice question -- slot 1 in this case. While slot 2 contains the answer that is correct for the other randomization choice.

**Example 3.2 – Placing a Variable in the Answer Field**

There are 8 recognized phases of the moon and we can create a randomized question that has 8 answer choices and asks about any of these 8 phases. This added level of complexity would quickly cause the approach from Example 3.1 to become cumbersome as more answer choices are possible. However, it becomes manageable by creating a variable that specifies the correct answer – so any of the answer choices will be able to be the correct answer and we simply need to identify the correct one with code.

The text of the question will remain the same, but now the lower half of the multiple choice question field becomes:

![Figure 3-3 – A Randomized Multiple Choice Question Text shown in Source Mode](image)

Note the Add Choices button at the top of the screen that allows you to add answer choices 5, 6, and 7. If you do mistakenly create an answer field that you don’t need, simply leave it blank and it will not be saved (there is no other way to get rid of it). Note that the first answer choice is selected and there is no way to unselect it (except by choosing another answer choice) but it can be edited out of the Correct Answer field after the question has been saved and reopened for editing.
The algorithm field now becomes:

```
$num1=rint(8);
$moonPhase = switch($num1,"new", "waxing crescent", "first quarter", "waxing gibbous", "full moon", "waning gibbous", "last quarter", "waning gibbous");
$answer = $num1 + 1;
```

![Figure 3-4 – A More Complicated Algorithm Field](image)

In creating the question, the phase of the moon has been coordinated with the order of the answer choices. So the first phase that will be asked about has as its answer the first answer choice. There are certainly other ways that this could be done, but this seems to be the easiest and most logical!

Note that $num1 is a random integer between 0 and 7, so $answer is a random integer between 1 and 8 [many things in computer science (but not all) start counting with zero]. The MapleTA commands rint and switch start counting at zero, while the answer choices start counting with 1.

The extension possibilities that follow encourage you to explore these examples and further the possibilities for randomization. Please note that increasing the randomization possibilities is not always a positive thing. MapleTA has no (simple) commands for making sure that randomization possibilities are not repeated or for making sure that all possibilities are eventually selected. Nor does it have any capabilities for ensuring that all items in a pool of questions are eventually selected. For example, if one programmed an assignment (like example 3.2) and wanted two questions, they could come up in identical form (a disaster on an exam) or could ask about two very close phases like waxing crescent and first quarter (undesirable).

It might be preferable to have two questions – one asking about the first 4 phases of a lunar cycle and one the last 4 phases – and create assignments having one instantiation of each. The possibility of duplication would be eliminated and the possibility of questions with very closely related phases would be greatly reduced. Users must remember to balance individual question randomization and selection from a pool of questions in assignment creation appropriately for the topic area and application (and what is optimal is likely to vary with the situation).
Extensions

- Extend example 3.1 by randomizing some of the distractors (wrong answers). You could program the incorrect answers to be especially distracting for that particular question’s permutation.
- Add a third randomization choice to example 3.1 focusing on the fact that the full moon rises at 6 pm. Have the question randomly ask about the rising time of one of three phases. You will need to add a another variable $wrongChoice2 and have three choices for the switch statements.
- Redo example 3.1 by placing a variable in the answer field.
- Challenge question (extra credit ;-). Redo example 3.2 by randomly asking about when a certain phase is on the meridian (halfway across the sky) which occurs 6 hours after rising and when the phase sets which occurs 12 hours after rising. You will need a more complicated expression to determine the answer field that will now depend on two variables.
4 Randomized Text Substitution

Another way to use randomization in problems is to use simple, non answer-changing, text substitution to prevent students from memorizing scenarios. You may think of this as “pattern matching prevention”. If a static question is used with the same actor and action in the question; students can memorize what the problem is about and its answer without actually learning the material being tested over. This is especially useful for story problems, in order to keep the problem dynamic. In this question we will have several different actors doing different things that have no real bearing on the answer, something that forces students to find the relevant information within the question.

We do think of this is a powerful technique for “keeping students focused on the concepts”. However, we encourage users to focus on the more important aspects of randomization related to question content.

After completing this tutorial you should…

- Be able to use random components to create variety in a question’s scene

New Functions

No new functions are introduced in this chapter.

Getting Started with Randomizing the Scene

One can think about creating the “scene” in terms of simple randomization. Here are simply hard coded some possibilities for the person’s name, mode of transportation, and time (the content of the question that would follow is really immaterial).

![Figure 4-1 – Randomizing the Scene](image)

Example 4.1 – Extensively Randomizing the Scene

Now examine a more complicated usage method. The meat of our question will be asking about moon phases again. I have modeled the question below, identifying the randomized components and their possibilities in brackets.

“[John/Jane] is [riding/driving/flying] [his/her] [bike/car/jetpack] [to/from] work at [time] and sees the moon directly overhead. What phase is the moon in?”
The only part that is associated with our answer is going to be the [time] component of the question above (First Quarter is overhead at 6pm and Last Quarter is overhead at 6am). Every other variable is merely a distractor.

It is also important to see that not every component is individually randomized. For example, the subject and their pronoun are specifically linked, so that “John is flying her bike from work at 6am…” doesn’t really sound right in several different ways. So, we have the verb and vehicle linked to one random variable, while the name and pronoun is generated by another. Going to or from work is then determined by the time (at 6am we our subject is going to work, and at 6pm they are going home from work). The text for the algorithm is illustrated in in Figure 4-3, and the question text is Figure 4-4. Several dummy answers were also added in addition to the variable right and wrong answers.
Assignment

Identify the scene in the preceding example that is generated when $\text{num1} = 1$, $\text{num2} = 2$, and $\text{num3} = 5$.

Scene = ___________________________________________________________________________________

Extentions

- Modify the question so the last sentence “What phase is the moon in?” references our subject again. That is the question should now be something similar to: “In what phase is he observing the moon?” (Hint: you should only need to modify the question text.)
- Change the question so where our subject is going to (or from) varies independently of the time. This can be done in several different ways.
5 Randomized Images in Multiple Choice

For this chapter, let’s begin by selecting Questions - New Question. Select Multiple Choice for the Question Type and enter The Question Description of Randomized Images. Our question will simply extend the concepts from previous multiple choice question chapters to handle the randomized display of images.

Our example will make use of a set of seven map images of continents similar to the figure of Africa shown to the right. It is assumed that these images are stored on an external server that we describe here with the variable $path, but the procedure of including a random component in a URL to the image is the same regardless of where they are stored. Note that the images are named continentX where X represents an integer from 1 to 7 (1=Africa, 2=Antarctica, 3=Asia, 4=Europe, 5=North America, 6=Oceania, 7=South America).

After completing this tutorial you should…

- Be able to insert images into MapleTA questions
- Be able to randomize what image is displayed

New Functions

strcat(“$a”, “$b”, “$c”) combines strings to form $a$b$c

Example 5.1 – Randomizing URLs of Images

The entire URL will be assembled with the MapleTA strcat command:

```
$num1 = rint(7) + 1;
$path = "http://cse.unl.edu/~klee/edu/maps/continent";
$URL = strcat(“$path”, "$num1", " .gif"));
$answer = $num1;
```

Note that rint is a random integer between 0 and 6, so $num1 is a random integer between 1 and 7. So while we named our images starting with 1, many things in computer science (but not all) start counting with zero. The MapleTA commands rint and switch start counting at zero, while the answer choices start counting with 1.

We can then use the $URL variable in our question in an <img src> tag.
Figure 5-1 - Randomized Images Algorithm Field

Figure 5-2 - Randomized Images Question Text (Source View)
Even though we could have entered source mode and typed in `<img src="$URL" align="right">`, we have instead made use of the insert image icon (which has an arrow pointing to it in Figure 5.2). This automatically inputs the html code `<IMG style="FLOAT: right" alt="" src="$URL"> for the user. You can make use of the Source button to see that our variable $URL is being inserted into the HTML tag which displays the image. Note that the image will not appear in the question editor – its position is indicated by a black “image not found” box. Once the question is previewed it will be shown with one of the 7 images selected randomly.

![Randomized Image Possible Answers](image.png)

Once again there are five multiple choice answers by default and we have used the Add Choices button to add two more fields. We have entered the continent names in the same order as our images are numbered for simplicity.
Extentions

- Redo Example 5.1 and change the order of the continents in the Choices for the answers (imagine that you had hurriedly typed them in and didn’t get them properly coordinated with your image names). Modify the Algorithm by placing a switch statement in the command defining the answer to properly accommodate the order of the continents in your answer choices. Use the preview function to test that your new coding works properly.
6 Nested Statements and Conditionals

In this chapter, we will explore using multiple random variables to create a self-assembling problem with different components. This is useful for comparison questions, ordering questions and situations where one could have multiple inputs. In the previous questions we have assembled simple problems with 1:1 matching of the question and answer and are generally constrained to only having a handful of possible answers. Here we are going to create conditions within which a problem will work and randomly assemble the question from those conditions. Our example problem described here will have 40 different possible questions with only a few lines of carefully designed code.

It is presumed that you are familiar with the concepts in previous tutorials, primarily the functions switch() and rint().

After completing this tutorial you should…

- Understand nested switch() and if() statements
- Understand conditional statements
- Be able to use the “condition:” statement to carefully disallow some random number possibilities
- Create self-assembling comparison questions

New Functions

We will add to your toolbox of known functions by introducing some new functions and new processes. The use of these functions should be very familiar to programmers, but their syntax is different in MapleTA.

\[
\text{if(<conditional statement>, <arg1 if true>, <arg2 if false>)}
\]

- To most programmers the statement “If condition is not zero, then arg1, else arg2” should be familiar. This statement condenses the familiar syntax into three arguments in the if() function. The first argument is generally a conditional statement (which returns 1.0 or 0.0, explained below), the second argument is what happens if the conditional statement is non-zero, and the third argument is what happens when the conditional is zero.

Conditional statements

Most of the following statements have two inputs and compares them in varying ways. If the condition is true, then they return 1.0; if it’s false they return 0.0. The logic equivalent operation is in parenthesis after the description.

\[
\text{ge(a,b) – greater than or equals to (a>=b)} \\
\text{le(a,b) – less than or equals to (a<=b)} \\
\text{eq(a,b) – equals to (a==b)} \\
\text{ne(a,b) – not equals to (a!=b)} \\
\text{lt(a,b) – less than (a<b)} \\
\text{gt(a,b) – greater than (a>b)} \\
\text{not(a) – not modifier (a==0)}
\]

Conditional Statements Tip

Since variables in MapleTA are untyped, all of the logic done purely with the idea that the number 0 is false, and anything else is true. This can be used creatively to simplify code.
The “condition:” statement functions as a gate on the algorithm and forces an overall prerequisite for evaluating the algorithm. If the condition is not met, then the algorithm starts over and reruns until the condition is met. In the following example, $a$ and $b$ can never be the same. Any conditional statement could be used in place of $\text{ne}()$, and any randomization function can be used in place of $\text{rint}()$ (such as $\text{range}()$ or $\text{rand}()$). If you create a situation where the condition can never be met, the algorithm will return a syntax error and not allow you to save.

```maple
$a = \text{rint}(5);
$b = \text{rint}(5);
\text{condition: } \text{ne}(\$a, \$b);
```

**Nested Statements**

Many statements can be nested to create more complex expressions by implementing logic in MapleTA. Look at the following example where $\$num1=\text{rint}(2)$ and $\$num2=\text{rint}(3)$:

```maple
\$variable1 = \text{switch}(\$num1, \text{switch}(\$num2, "This is 0-0", "This is 0-1", "This is 0-2"),
             \text{if}(\text{eq}(\$num2, 0), "This is 1-0", \text{if}(\text{eq}(\$num2, 1), "This is 1-1", "This is 1-2")));
```

This begins to have lots of parenthesis and other things to keep track of, so I have highlighted related sections. Blue is the arguments to the ‘big’ switch statement. The second blue argument is just another switch statement (which you should be familiar with reading), and the third blue argument is actually a pair of nested ifs for which I have underlined each argument. The orange if has another if statement in its else argument (which is divided by yellow underlines).

Ultimately our statement above is equivalent to $\$variable1 = “This is $\$num1-$\$num2”, but this helps to illustrate nesting statements before we move to an application where we use nested statements.

**Comparison Questions Using Logic**

We are going to create a question that compares the sizes of astronomical objects. We have 5 options with their indices in parenthesis (ordered from smallest to largest): a moon (0), a planet (1), a star (2), a solar system (3), and a galaxy (4). The random number associated with each is proportional to their size, so we can compare sizes using these indices later on. Finally, we are going to ask two different types of questions after we determine the subjects: are they larger than (0) or smaller than (1) each other.
The variables we are going to use and define are:

$choice1, \, $choice2, \ldots, \, $choice5 – the text associated with our objects

$fb1, \, $fb2, \ldots, \, $fb5 – our case specific feedback text

$num1, \, $num2 – the index of our objects (with the condition that ne($num1,$num2))

$num3 – our comparison choice (larger (0) or smaller (1))

$part1, \, $part2 – the text from $choice variables above related to our random numbers

$equality – the text related to $num3

$probttext – our problem text passed to the question (rather than coordinating several variables in the question designer, we will assemble the entire question as a variable using strcat())

$answer – our answer’s text (set using comparison logic of the above choices)

$notanswer – the incorrect answer’s text (set using comparison logic of the above choices)

$feedbacktext – we are also going to assemble feedback text. This variable is entered into the feedback field, and will show the student context specific explanations.

Using Variables Tip

The text in the $choice variables could be integrated directly into the select() statement in $part1, $part2, and $feedbacktext without issue – but this allows us to more easily use this question as a template in the future. We can just replace the value of the choice variables and every instance of this text later on is replaced. This is good programming practice as we use the same text multiple times later on. Further, it helps to simplify the switch() statement to avoid complicating it further.
Edit the code for your algorithm in the text box below, or click "Show Designer" to use the algorithm designer. The algorithm designer tool allows you to define algorithms for your question by completing a form.

```
$choice1 = "a moon";
$choice2 = "a planet";
$choice3 = "a star";
$choice4 = "a solar system";
$choice5 = "a galaxy";

$fb1 = "The Earth's moon has a diameter of about 2500km. It is one of the largest satellites belonging to a planet in the solar system.";
$fb2 = "Planets vary in size. In our solar system Mercury is the smallest with a diameter of about 5,000km and Jupiter is the largest with a diameter of 140,000km.";
$fb3 = "Stars vary in size. Our Sun is an average star with a diameter of about 1.4 million km.";
$fb4 = "Solar systems are intrinsically larger than stars since they encompass a star in addition to its constituent planets/other objects. Solar systems are on the order of billions of km in size.";
$fb5 = "Galaxies are made up of billions of stars and are on the order of thousands of light years across.";

$num1 = zint(5);
$num2 = zint(5);
condition: ne($num1,$num2);
$num3 = zint(2);

$part1 = switch($num1,"$choice1","$choice2","$choice3","$choice4","$choice5");
$part2 = switch($num2,"$choice1","$choice2","$choice3","$choice4","$choice5");
$equality = switch($num3,"larger than","smaller than")

$protext = strcat("Is ","$part1"," equality ","$part2","\?"\);

$answer = if(eq($num1,1),if(le($num1,$num2),"Yes","No"),if(gt($num1,$num2),"Yes","No"));
$notanswer = if(eq($num1,1),if(le($num1,$num2),"No","Yes"),if(gt($num1,$num2),"No","Yes"));

$protext = strcat("So ","$part1",if(gt($num1,$num2)," larger than "," is not larger than "),"$part2."));
```

---

Figure 6-2 - Algorithm Text

---

Figure 6-1 - Question Text

MapleTA Randomization Manual
Nearly all of the work is done in the algorithm field; and we simply insert our few variables into the question text box, feedback and response areas. This can be done as a multiple choice question without changing the algorithm. Just make sure to enter the algorithm first, so the variables are defined for their first entry into the question text field.

**Adding Same Rank Objects**

We can further customize the problem by changing some of the variables. For instance, if we presumed that moons and dwarf planets were the same rank for this question, both could be included at the same rank ($\text{num1}$ or $\text{num2}$). We can modify our variable for the moon to be: $\text{choice1} = \text{switch(\text{num1},"a moon","a dwarf planet")}$; to set our lowest rank choice equal to one of those two options. We use $\text{num1}$ in this choice to allow us to customize the feedback later (num = choice’s number). We will do the same with $\text{choice4} = \text{switch(\text{num4},"a solar system","a planetary nebula")}$;

The variables we have added or changed to allow for multiple same-rank options are described below. Our use of variables for the text in our question helps make this transition much easier as we only have to change a few lines to add an extra layer to our question. The question text field does not change.

*Added: $\text{num1}, $\text{num4}$ – These are our random components to choose the new options

*Changed: $\text{choice1}, $\text{choice4}$ – We added a switch statement like described in the previous paragraph to give us multiple text options at the same rank.

*Added: $\text{fb1b}, $\text{fb4b}$ – These are the new feedback strings to be displayed for a dwarf planet and a planetary nebula.

*Changed: $\text{feedbacktext}$ – We also changed the feedback text to include the new options. Where there was a “$\text{fb1}$” in the previous example we now have a switch($\text{num1},"\text{fb1","fb1b"}$) to direct that entry to the correct feedback statement.

We avoid some complications and ambiguities in the question automatically by only comparing different ranks and changing our comparison once we have determined the ranks to compare. Comparing same rank objects requires a different method and additional variables.

**Extensions**

- Create a new ranking problem that instead asks “Compare object A and B” with possible responses of “A is larger than B”, “A is smaller than B”, and “A is similar size to B”. Take special note that this can be customizable with any sort of ‘scale’ – it doesn’t have to be just sizes. For instance one might ask about geological timescales where there can be some overlap due to the hierarchy. The Jurassic period came after the Triassic period, but both are part of the greater Mesozoic era.
Figure 6-3 - Algorithm Field with Multiple Same-Rank Options
7 Random Multiple Selection

The Multiple Selection question type is basically a multiple choice question where several answers are correct and required (and there are usually several wrong answers as well). This type of question is useful for exploring the characteristics of something, membership in a group, achievements of a person, occurrences during a certain time period, etc. Here there are a list of things that apply to a certain criteria.

This type of question is especially powerful when it randomly varies the criteria that students select things that apply to it. Like other randomized problems, we will have a predetermined number of topics we want to ask about using the same question framework. Multiple selection is an extremely powerful question type that can be used in many different ways.

After completing this tutorial you should…

- Be able to create a randomized multiple selection question.

New Functions

There are no new functions in this tutorial.

Example 7-1A: Types of Critters

As a simple example to motivate use of this question type, let’s create the question “Identify the _____ from the following list:” where the blank can be one of three types of creatures: mammals, fish or insects. In planning for this question type, it is useful to create a table of creature types and the possible answers (shown to the right). This is a simplistic example to direct focus on the method used for question creation rather than worrying about what the question is asking. For any question with multiple answers, careful planning is highly recommended.

Let’s first take a very simple approach. Lets create new question – of type Multiple Selection – and describe it as MS_MammalsFishInsects. We first need to create any variables that we plan on using in the question in the algorithm field. In the algorithm field let’s enter the following 3 variables:

```plaintext
$num = rint(3);
$cat=switch($num,"mammals","fish","insects");
$answer=switch($num,"1,2,3,4","5,6,7","8,9");
```

Make sure that you click save which will return you to the opening page and you may then click Add Choices 4 times. You can then simply enter the critters sequentially in the choices fields.
A preview of the question and the Partial Grading Explained message are shown below.

**Question:**
Which of the following are mammals?

- trout
- salmon
- cat
- horse
- mouse
- dog
- beetle
- fly
- shark

Questions that are partially graded calculate their grade according to the following formula:

\[
\text{Grade} = \frac{\text{# correct choices} - \text{# incorrect choices}}{\text{# correct answers}}
\]
Example 7-1B: Types of Critters 2

Let’s now look at that same example again in a longer more exhaustive way that actually creates tremendous flexibility and adaptability in the long run. It now becomes pretty simple to copy this question and create a new question from this question’s code.

Variables Used

The variable names have been chosen to be descriptive. These should be created in the algorithm field before entering the question.

$\textit{num} – \text{Our primary question index}
$\textit{cat1}, \textit{cat2}, \textit{cat3} – \text{The text of the various categories. (Mammals, Fish and Insects)}
$\textit{choice1}, \textit{choice2}, \ldots, \textit{choice9} – \text{The text of my choices (Cat, Dog, Horse, Trout, Fly…)}.
$\textit{numtext} – \text{This variable is used to fill in the blank for “Identify the _____ from the following list:”}
$\textit{answer} – \text{This is sent to the answer field to indicate the correct choices based on our $\textit{num}$.}$
$\textit{fb1}, \textit{fb2}, \textit{fb3} – \text{This is our feedback text for each case.}$
$\textit{feedbacktext} – \text{This is case specific feedback text which is displayed after a question attempt.}$

Algorithm

The algorithm starts on Line 1 by randomly picking which category we will use. The next several lines define our categories, the answer options, and their associated feedback text. Each of the variables starting with $\textit{choice}$ will be added to the possible answer fields for the Multiple Choice question. We could just put in each of the options in the answer fields (and not use variables), but doing it this way allows for easily editing and creating similar questions (and makes tracking the index of each answer easier). In the last few lines we set $\textit{numtext}$ to the proper category. We will insert this variable into the question. Again, the category choices could be written into the switch statement, but using variables allows for easy editing and organization. We also define which indexes are the correct answers based on our category selection. Note that the numbers listed correspond to the selection number as they are shown in the Multiple Choice list (this is where $\textit{choice#}$ is more helpful).
than just putting the names directly into the Multiple Choice question – we can relate the indexes to the variable name and do not need to separately track what order we put the choices into the Multiple Choice question).

![Figure 7-4 - Random Multiple Selection Question Text](image)

The variable $answer should be entered in the algorithmic answer field (available after hitting ‘OK’ to the Edit Response Area dialog). Also, enter the variable $feedbacktext into the feedback field it to give the student appropriate post-question dialog.

**Extensions**

- Adapt the question to instead find animals NOT of two categories listed by asking “What animals don’t belong to x family and y family?” (Hint: There is a verbose method and an easy method.)
Example 7-2: Contribution of famous 20th Century Astronomers

Let’s look at a second example focusing on the contributions of Harlow Shapley and Edwin Hubble – the two most famous astronomers of the 20th century. We will basically provide a list of the major accomplishments of both and several distractors and ask students to identify which are the major contributions of one or the other (selected randomly).

We first identify variables in the algorithm:

\[
\text{num1} = \text{rint}(2); \\
\text{astroguy} = \text{switch}($\text{num1}$, "Harlow Shapley", "Edwin Hubble"); \\
\text{answer} = \text{switch}($\text{num1}$, "1,5", "2,3,4");
\]

As has been mentioned previously, it may be easier to simply create “dummy” variables. You probably wouldn’t know the answer values until you have entered your choices (unless your planning was quite meticulous). We have also typed up a feedback paragraph. This is a generic chunk of feedback designed to work whether it is Shapley or Hubble that is asked about.

![Figure 7-5 – Famous Astronomer Feedback](image)

We can now enter the question (with the astronomy that is being asked about as a randomized variable),

![Figure 7-6 – Famous Astronomer Question](image)
the possible answer choices,

<table>
<thead>
<tr>
<th>Possible Accomplishments</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimated the size of the Milky Way by finding the distance to the center</td>
</tr>
<tr>
<td>observed and categorized many other galaxies</td>
</tr>
<tr>
<td>used a Cepheid to find the distance to the Andromeda Galaxy</td>
</tr>
<tr>
<td>stated a simple rule for relating a galaxy’s velocity of recession to its distance</td>
</tr>
<tr>
<td>mistakenly thought “spiral nebulae” were in our own galaxy</td>
</tr>
<tr>
<td>expressed a relationship between a Cepheid’s luminosity and its period of pulsation</td>
</tr>
<tr>
<td>showed that dark matter is abundant in the outer parts of the Milky Way</td>
</tr>
<tr>
<td>carefully described the differences between globular and open clusters</td>
</tr>
</tbody>
</table>

(In addition to the choices selected above, you can also add algorithmic values as correct answers.)

Algorithmic Value Answers:

Change the order of the choices?
- Yes
- No

Allow more than one selection?
- Yes
- No

Figure 7-7 – Famous Astronomer Possible Accomplishments

and the final question appears as:

<table>
<thead>
<tr>
<th>Question:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which of the following describe the astronomical work done by Edwin Hubble?</td>
</tr>
<tr>
<td>stated a simple rule for relating a galaxy’s velocity of recession to its distance</td>
</tr>
<tr>
<td>carefully described the differences between globular and open clusters</td>
</tr>
<tr>
<td>used a Cepheid to find the distance to the Andromeda Galaxy</td>
</tr>
<tr>
<td>expressed a relationship between a Cepheid’s luminosity and its period of pulsation</td>
</tr>
<tr>
<td>showed that dark matter is abundant in the outer parts of the Milky Way</td>
</tr>
<tr>
<td>observed and categorized many other galaxies</td>
</tr>
<tr>
<td>mistakenly thought “spiral nebulae” were in our own galaxy</td>
</tr>
<tr>
<td>estimated the size of the Milky Way by finding the distance to the center</td>
</tr>
</tbody>
</table>

Partial Grading Explained

Figure 7-8 – Final Form of Randomized Famous Astronomer Multiple Selection Question
Example 7-3: Thermal Expansion

Our final problem is much more complicated mathematically as it deals with how much different metals (lead, aluminum, brass, copper, and steel) expand when their temperature rises.

\[ \Delta L = \alpha L_0 \Delta T \]

Students are asked to use the equation above that reads: the change in length is equal to \( \alpha \) the coefficient of thermal expansion – times \( L_0 \), the original length – times the change in temperature. We will randomly select a final temperature that is either far below or far above the original temperature and provide a static original length (but that could be randomly generated as well). Students are asked to select all of the metals where the change in length is either 3 cm, 4 cm, 5 cm, 6 cm, or 7 cm greater or less than the starting temperature and in this problem we will leave one of the possibilities as having a “null” answer. We will include the coefficients of thermal expansion for each of the 5 metals in the choice of that metal, which certainly isn’t the only way in which this question could be formatted. Realize that this problem has been carefully engineered so that the answers come out this manner.

```plaintext
$num1=rint(2);
$temp2=switch($num1,"-196","300");
$verb=switch($num1,"cooled in liquid nitrogen","heated in a furnace");
$action=switch($num1,"lowered","raised");
$num2=rint(5);
$change=switch($num2,"3.00","4.00","5.00","6.00","7.00");
$num3=rint(2);
$change2=switch($num3,"greater","less");
$ans=switch($num1,switch($num3,switch($num2,"1,2,3,4","1,2,3","1,2","1","" ",switch($num2,"5","4,5","3,4,5","2,3,4,5","1,2,3,4,5")
 switch($num3,switch($num2,"1,2,3,4,5","1,2,3,4","1,2,3","1,2","1")
switch($num2,"","5","4,5","3,4,5","2,3,4,5");)

$alpha1=29E-6;
$alpha2=24E-6;
$alpha3=19E-6;
$alpha4=17E-6;
$alpha5=12E-6;
$length=10.0;
$temp1=23.0;
$DeltaT=abs($temp2-$temp1);
$DeltaL1=abs($alpha1*$length*(Stem2-$temp1)*100);
$DeltaL2=abs($alpha2*$length*(Stem2-$temp1)*100);
$DeltaL3=abs($alpha3*$length*(Stem2-$temp1)*100);
$DeltaL4=abs($alpha4*$length*(Stem2-$temp1)*100);
$DeltaL5=abs($alpha5*$length*(Stem2-$temp1)*100);
```

Figure 7-9 – Variables for the Thermal Expansion Multiple Selection Question
Figure 7-10 – Answer Choices for the Thermal Expansion Multiple Selection Question

Figure 7-11 – Final Version of the Thermal Expansion Multiple Selection Question
8 Introduction to Numeric Questions

This chapter will provide a basic tutorial on Numeric Questions covering the commands for random number generation and trigonometric functions as well as grading concerns. This question type is appropriate for a standard problem in introductory science classes where students are asked to plug into a formula and calculate an answer. As was the case with Multiple Choice questions, numeric questions can also be created as part of a larger question in the question designer. This approach will be covered in later sections, as its benefits also bring increased complexity.

We have already seen the rint() command in previous sections. This section will introduce the commonly used range and rand commands as well as the sig and decimal commands which have their uses.

<table>
<thead>
<tr>
<th>Numerical Randomization Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>range: returns a random integer.</td>
</tr>
<tr>
<td>range(n) – generates a random integer between 1 and n</td>
</tr>
<tr>
<td>range(m,n) – generates a random integer between m and n inclusive</td>
</tr>
<tr>
<td>range(m,n,k) – generates a random integer between n, n+k, n+2k, …, n</td>
</tr>
<tr>
<td>rand: returns a random real number</td>
</tr>
<tr>
<td>rand(m,n) – returns a random real number between m and n inclusive</td>
</tr>
<tr>
<td>rand(m,n,k) – returns a random real number between m and n to k significant figures</td>
</tr>
<tr>
<td>sig: controls the number of significant figures displayed in an expression</td>
</tr>
<tr>
<td>sig(n,x) – returns the expression x rounded to n decimal places</td>
</tr>
<tr>
<td>decimal: controls the number of decimal places displayed in an expression</td>
</tr>
<tr>
<td>decimal(n,x) – returns a number rounded to n decimal places</td>
</tr>
</tbody>
</table>

The table above reflects the usage of these commands as described in the MapleTA manual. Note that it describes range as only being applicable for randomizing integers. For many years, we have used it for real numbers as well. And it still does work – in fact, it has some functionality that is difficult to duplicate using these existing commands as documented. For example, if one generates a random real number of “42.0” which has 3 significant figures, it will be displayed by MapleTA as “42” making one think it only has 2 significant figures – trailing zeroes are truncated. It is easy to avoid this using the range command on real numbers. For example range(31.1,49.9, 0.2) will never generate any real number ending in “.0” and avoids this possibility. There is no documented “step” command for reals numbers. There are more complicated ways to avoid significant figure ambiguities, but we have chosen to make use of the range function on real numbers (not covered in MapleTA documentation) since we have been doing that for years and it can be done quite simply.
MapleTA has all of the standard trig functions [sin(), cos(), tan(), csc(), etc.] as well as their inverse functions [arcsin(), arccos(), arctan(), arccsc() etc.] The arguments for trig functions are assumed to be in radians and the inverse functions return the angles in radians. Thus, conversion between degrees and radians is very often necessary.

MapleTA offers a wide variety of grading options. We encourage instructors to consider how infuriating it is for students to have answers marked as incorrect for syntax errors regarding units, rounding errors, or significant figures. Thus, we typically suggest that instructors specify a grading system, pay no attention to significant figures, and provide direction regarding the desired units for an answer. However, instructors have total control over how they want to handle these issues and MapleTA has become far more tolerant of problems with answer entry and helpful over time.

Let’s begin by creating a simple vector addition problem that could occur in an introductory physics course. Begin by selecting Questions - New Question. Choose The Question Type to be Numeric and enter The Question Description of Simple Displacement. This question will have a displacement from the origin, a second displacement in another direction, and ask students to calculate the direction of the final displacement from the origin. Thus, our question will make use of 3 randomized variables and need a corresponding (randomized) answer and I will begin by setting these up with temporary values in the algorithm field and saving their values. ($disp1 = 1; $disp2 = 1; $angle = 1; $answer = 1;)

We can now enter our question – click the Next button.

![Figure 8-2 – Simple Displacement Question](image-url)
Note that the $answer will eventually be defined correctly in the algorithm field. It may seem a little clumsy to specify the use of a certain unit the way that we have done. One will see in later chapters that including a numeric question as part of an inline question allows for more convenient options.

Figure 8-3 – Simple Displacement Question Answer

There are a number of options for specifying the required precision of the answer. The easiest way to handle this is to set a certain percentage – say 3% which works well for most problems -- and MapleTA will mark all answers as fully correct that are within this tolerance of the given answer. For a science course accepting scientific notation is appropriate, other options are probably not necessary.

Specify precision:

- **Require absolute accuracy**
- **# Figures:** [1]
  - The student answer must be presented with exactly this number of significant figures.
- **Accept +/- err:**
  - E.g. To accept any number between 2.74 and 2.76 set err to be 0.01
- **Accept +/- k in n-th place:**
  - E.g. Set k to 2 and n to 3 to accept only 2.73, 2.74, 2.75, 2.76 and 2.77.
- **Accept +/- perc%:**
  - E.g. To accept any number within 10% of the correct answer, enter 10 for perc

Specify input format:

- **Accept 1000 separator**
- **Accept scientific notation**
- **Accept $ signs**
- **Accept arithmetic**

Select the style of negative numbers you want to accept:

- $-1.234.0$; $-1.234E+3$ or $(1,234.0); (1.234E+3)$
- $-1.234.0$; $-1.234E+3$
- $(1,234.0); (1.234E+3)$

Figure 8-4 – Specifying the Requirements for a Numeric Answer

We can now flesh out the algorithm field by setting realistic value possibilities for $disp1$, $disp2$, $angle$, and calculate the final answer $answer$. 

MapleTA Randomization Manual
Note that we multiply the angle by $\pi/180.0$ to convert from degrees to radians before doing any of the trig calculations. All calculations involving angles – such as determining the net $x$ and $y$ displacements – is done in radians. Once we have determined the final value of the angle with the arctan function we multiply by $180.0/\pi$ to convert our final answer from radians to degrees.

![Figure 8-5 – Simple Displacement Question Algorithm](image)

The possible values of the variables can be repeatedly generated by clicking the Refresh button. This allows the user to gain confidence that the randomization is actually having the effect that is desired. If one clicks on the Units link (as students can do), you will get a list of recognized units. The acceptable choices pertinent to this problem are deg, degree, degrees, rad, radian, and radians.

![Figure 8-6 – Final Form of Displacement Question](image)

We can also program our randomized problem to display appropriate randomized feedback to students. Enter the feedback mode and we will create solution that will appear to students once they click grade. Note that we want to calculate the total $x$ and $y$ components of the displacement and then use the arctan function to calculate the angle. The last part of this – the arctan function – will require MathML to typeset the equation and we will ultimately insert our variable values into the MathML.

Use the “large sigma” icon at the right end of the bottom row of the question editor to launch the equation editor. There are often some security permissions to authorize to get this java program up and running. After that has been accomplished, you can use the menu of palettes to add mathematical symbols and structures such as superscripts and square roots. You are encouraged to explore the
menus and gain a feeling for what is present and where it is located. The only way to master the
equation editor is through experimentation. Note that the second arctan has been cut-and-pasted.
When this feedback is actually displayed by MapleTA it will insert the current values for $\text{totx}$, $\text{toty}$,
and $\text{answer}$ as defined in the algorithm field.

![Equation Editor](image)

**Figure 8-7 – Creating Feedback using the Equation Editor**

<table>
<thead>
<tr>
<th>Description:</th>
<th>Simple Displacement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade:</td>
<td>1.0</td>
</tr>
<tr>
<td>You drive directly east for 129 mi and then drive 189 mi at an angle 43.5° north of east. What is the direction of your net displacement from the starting position? (It is assumed that your final answer is in degrees north of east and you should simply state &quot;deg&quot; as your unit.)</td>
<td></td>
</tr>
<tr>
<td>Your Answer:</td>
<td>26.05 deg</td>
</tr>
<tr>
<td>Correct Answer:</td>
<td>26.1 deg ± 3.0%</td>
</tr>
<tr>
<td>Comment: The problem requires us to combine two displacement vectors and determine the direction of the total displacement from the starting position. One should determine the $x$ and $y$ components of the individual displacements $D_1$ and $D_2$ and add them to identify the components of the resultant $R$. $R_x = D_1 + D_2 \cos \theta$ $R_y = 0 + D_2 \sin \theta$ $\theta = \tan \left( \frac{R_y}{R_x} \right) = \tan \left( \frac{139.1}{266.1} \right) = 26.1^\circ$</td>
<td></td>
</tr>
</tbody>
</table>

![Feedback](image)

**Figure 8-8 – Final Appearance of Feedback**
9 Limiting Randomization with Condition Statements

Condition statements are a method for controlling the possibilities of randomization. In physics problems, they are often used to keep “unphysical” things from occurring. It should be noted that they really don’t supply anything new as far as capabilities, but they allow you to be confident of the outcome of a set of algorithmic operations on randomized variables in a numeric problem with far less effort than would otherwise be possible.

Example 9.1 – Inclined Atwood’s Machine

Here we will explore a common physics problem – two masses connected by a rope over a pulley on an inclined plane – to motivate the use and necessity of condition statements. This problem can get rather complicated, so I will focus on discussing some of challenges when coding it in MapleTA and gloss over the physical concepts involved. What is typically asked of students is: (Assuming the masses are originally held still and then released), what is the acceleration of the two masses on the frictionless incline? This requires students to apply Newton’s second law and understand that the two masses will have the same acceleration since they are connected. The direction of motion is determined by which is larger – the weight of mass of M1 turning the pulley counterclockwise or the component of the weight of mass M2 down the incline turning the pulley clockwise.

\[ \Sigma F_{\text{pulley clockwise is positive}} = M_2 g \sin \theta - M_1 g = (M_1 + M_2) a \]

\[ a = \frac{M_2 g \sin \theta - M_1 g}{M_1 + M_2} \]

If \( M_2 g \sin \theta > M_1 g \) then the block M2 moves down the incline and acceleration is positive. If \( M_2 g \sin \theta < M_1 g \) then the block M2 moves up the incline and acceleration is negative. A given situation can be positive or negative depending upon which way you point your coordinate system – and either choice made by a student is correct. There are several ways to head off this ambiguity. One way is to simply ask for the magnitude of the acceleration in the question. Another way is to spell out to students that “forces that cause the pulley to rotate clockwise are positive” in the question. A third way is to use a condition statement to limit the range of possibilities in the problem’s algorithm block (randomizing M1, M2, and \( \theta \)) and only allow the block to go down the incline. At the bottom of the algorithm block, you check with a condition statement that \( M_2 g \sin \theta \) is greater than \( M_1 g \) – and if it isn’t execution jumps back to the beginning of the block and starts over. So the question could become: what is the acceleration of M2 down the incline?

If we add friction \( \mu_k \) and \( \mu_s \) to the problem it greatly complicates things in that M2 can go down the incline, up the incline, or not move at all depending upon the values of M1, M2, \( \theta \), \( \mu_k \), and \( \mu_s \).
Example 9.2 – Horizontal Range Problem

Let’s look at a simple example making use of the Horizontal Range formula from Physics. This is used when you are launching a projectile at a given angle with a given initial velocity and want to know how far it travels before returning to the ground. It is a symmetric function about a starting angle of 45° — i.e. both 36° and 54° have the same horizontal range. Let devise a problem that requires students to use the formula and understand the symmetry.

We will create a new question of type Numeric and describe it by Horizontal Range. We will need to specify variables in the algorithm field for the horizontal range and both angles that would yield such a horizontal range — one that will be used in the calculation and one that will be the answer. Then we can randomize between them — giving either an angle above 45° or below — which requires some complex logic.

The last statement in the randomization is “condition: ne($angle0,45.0);”’. This command basically prevents the initial angle from being 45.0°, which would be the maximum possible horizontal range for that initial velocity and there is no symmetric angle. Admittedly, that doesn’t happen very often and there are other ways of coding this problem to keep that from happening than using the condition statement. Yet there are more complex situations where the condition statement is the easiest quickest way to prevent an unfortunate randomization possibility. Note that if $angle0 did equal 45.0°, the entire algorithm code is re-executed from the top. It is possible to create a condition statement that can never (or very infrequently) allow the algorithm loop to execute and MapleTA will alert you to this situation.

![Horizontal Range Algorithm Variables](image-url)
The question is shown in the text editor:

A football is kicked and given an initial velocity of \( v_0 \) m/s at a certain angle with respect to the horizontal. It lands \( H \) m away. At what other \( \theta \) angle would the football travel the same distance?
The feedback window is shown below:

Figure 9-4 – Horizontal Range Question Feedback

At this point, it was realized that some additional variables $\text{comp1}$ and $\text{comp2}$ would be useful and the following code was added to the algorithm window;

$\text{num1}=\text{if(}\text{lt}($\text{angle0},45),0,1);$

$\text{comp1}=\text{switch($num1,"less than","greater than");}$

$\text{comp2}=\text{switch($num1,"greater than","less than");}$

The final question in the preview window after grading is shown below:
Description: **Horizontal Range**

Grade: 1.0

A football is kicked and given an initial velocity of 43.8 m/s at a certain angle with respect to the horizontal. It lands 168.7 m away. At what other smaller angle would the football travel the same distance?

**Your Answer:** 29.8 deg

**Correct Answer:** 29.8 deg ± 3.0%

**Comment:** Note that the horizontal range function is symmetric about an angle of 45°, which yields the maximum horizontal range for a given initial velocity. Thus, we should first calculate this first angle.

\[ R = \frac{v_0^2 \sin(2\theta)}{g} \]

\[ \theta = \frac{1}{2} \sin^{-1} \left( \frac{gR}{v_0^2} \right) = \frac{1}{2} \sin^{-1} \left[ \frac{\left( \frac{9.81 \text{ m/s}^2}{2} \right)(168.7 \text{ m})}{\left( \frac{43.8 \text{ m}}{2} \right)^2} \right] = 60.2 \degree \]

Note that this angle is greater than 45°, so the second angle must the less than 45° by the same amount. So the other angle is 29.8°.
10 Overview of Question Designer

In a simple question type (such as Multiple Choice), one is limited to a single question with a single response. We can randomize the values in this single question to make all aspects of it dynamic, but it is still a single question. MapleTA has the functionality to create a question with multiple response types combined into one question in what is called the Question Designer (referred to as an inline question in the past). A Question Designer question receives one score for the entire question, but you control the weighting of the sub-parts. The Question Designer allows us to ask about multiple related things at once and probe student understanding at a deep level through follow-up questions. Note that all sub-parts of a Question Designer question “share” the same set of algorithmic variables.

The Question Designer is incredibly flexible – you can make assessment materials in any format that you desire. You should endeavor to slowly build skills with the question designer and ultimately create assessments that look like complete web pages with student response cells embedded throughout – forcing students to participate in the flow of information. This is incredibly useful for creating tutorials.

After completing this tutorial you should...

- Be able to use Question Designer to create a multiple response question

New Functions

abs(number) – returns the absolute value of the input number

---

Background on the Doppler Shift

The Doppler shift is the compression or extension of a wave (generally sound or light) due to a travelling source or observer. The most commonly used example of this is a moving vehicle’s horn: while a car is moving towards you, each peak of the sound wave is closer together so the frequency of the sound is higher (and the pitch goes up). As the car moves away from you, the frequency is decreased and the sound is lower. Observed light from astronomical objects behaves in the same way. When the separation between an object and us is increasing, the light is stretched and ‘redshifted’ (e.g. the light has a lower than expected frequency, so the light is redder than expected) and when an object’s separation is decreasing the light is considered to be ‘blueshifted’ (e.g. the light has a higher than expected frequency and is bluer than expected). Note that the Doppler Shift only acts along the radial direction – along the line of sight.

\[
\frac{\lambda_{obs} - \lambda_{lab}}{\lambda_{lab}} = \frac{v_{radial}}{c}
\]

The Doppler Shift formula above relates the observed value of wavelength of light $\lambda_{obs}$ to that when the source and observer are not moving $\lambda_{lab}$ to the radial velocity (how the separation between source and observer are changing, it doesn’t matter here which is moving).
Example 10.1 – A Question Designer Doppler Shift Problem

The advantage to a multi-part question, especially when used in an assignment (exam, quiz, homework, etc.), is that ideas can be coupled together and related in a coherent way. While nearly any question we design using the Question Designer could be split into parts and asked individually, the benefit here is in relating multiple parts of the same problem that use the same set of randomized variables. In this example, we will be asking the student to solve a mathematical expression related to the Doppler shift and then have them analyze the answer qualitatively. This could have easily been two problems independent of each other, but we want to tie the concepts together and present them as a related pair of questions.

![Figure 100-1 - Question Designer Algorithm Field](image)

The code for the algorithm shown in Figure 10-1 is quite short. Note that an observed wavelength is generated. Half of the time it will be longer than the laboratory (non-moving) value of H\(\beta\) and half of the time it will be shorter. The variable $answer1$ stores the left-hand-side of the Doppler Shift formula – the speed of change in separation between source and observer in terms of the speed of light $c$. $answer2$ simply stores whether the source and observer are approaching each other or receding (which will be a Multiple Choice question regarding the direction of the observed object). To avoid any confusion due to potential small differences between the observed value and the expected value we join the possible ranges with a randomized switch statement to determine if we will use a number from the higher or lower range of random numbers.

Note that both answers are ultimately determined by what our random $observed$ value is and how it compares to the rest value of H\(\beta\). After the algorithm field is populated with our variables, we can then define how they are used in the question text. A real advantage of using the Question Designer is that we could actually input our variable names in the question text first and define them afterwards – this is something that cannot be done with the single question types. (Remember that this is an evolving area of MapleTA!)

This text box is almost identical to the Question Text fields from the other question types, with one notable exception: a button to Insert/Edit Response Area (indicated by the arrow). The Response Area object is placed in line with the cursor when it is selected and immediately a dialog comes up to design the Response Area. If you
need to edit the object after it has been placed, the Edit button (which doesn’t look like a button) on the
Response Area can be double clicked which will allow editing. The Response Area “object” can be manipulated
within the text to allow for extra spaces, lines, etc.

The dialog shown in Figure 10-3 has all of the options related to the numeric response area and the particular
configurations used in this exercise. The Units Part is left blank since we are not asking for units to this
question. The units field in the response area will not display to the student since we do not have any answer to
our Units field (and I have manually put a ‘c’ to indicate that the answer is a fraction of the speed of light after
the response area).
To input the multiple choice question, we move the cursor down the page and again click on the Insert/Edit Response Area button and select the multiple choice option. Thus, the Insert/Edit Response Area was clicked twice in this problem – once to create the numeric problem and one to create the multiple choice problem. It will need to be clicked once for each sub-part of the Questions Designer question. Note that the Edit label associated with either question part can always be double-clicked (although it looks rather innocuous) to bring up the dialog boxes controlling the question for further editing.

The dialog shown in Figure 10-4 has all of the options related to the multiple choice response area and the particular answer choices used in this exercise. Inputting a multiple choice response area is a two-step process. After we input our options and hit OK another dialog shown in Figure 10-5 gives us the opportunity to input feedback and choose our correct answer. We will populate the Multiple Choice answer field with the variable $answer2.

Note that we have weighted the two parts of our questions equally – they will each be graded for 50% of the total credit. We could have weighted the two parts quite differently.
Extensions

- Edit the question to randomly determine the hydrogen line being detected. For reference, H-alpha = 656.3 nm and H-gamma = 434.1 nm (H-beta is used in the problem). Hint: you will want to scale the random numbers accordingly, create a text value that is passed into the problem, and coordinate which line you are measuring throughout the problem.
- Add an additional multiple choice question asking students to identify whether the light was “redshifted” or “blueshifted”.

Figure 100-4 – Multiple Choice Response Area Dialog

Figure 100-5 – Multiple Choice Response Area Dialog
11 Narratives in the Question Designer

This chapter will explore “narratives” – predominantly text summaries in the question designer that are often used as reading quizzes. The simple text Response Area is officially referred to as a List. You can use a pull-down menu of choices and have the user select the correct answer or create a blank field where the user types the response where there are complicated options for grading like Regular Expression Match. There is no reason why lists cannot be intermixed with multiple choice, numeric, and essays.

Note that we will demonstrate the most straightforward approach – specifying a list of options in a pull-down menu. We will wish to randomize both the correct and incorrect answers for a list question, but note that it is not possible to randomize the weights. Thus, the correct answer must always go in the same item slot with the full weight even if randomization changes its value. Thus it is often necessary to randomize both the correct and incorrect answers in some coordinated fashion to make this procedure work.

Users may find the two weights confusing. Note that the first entitled “Weighting” describes the maximum percentage obtainable for that response area – that particular value of weighting divided by the sum of all of the response area weightings. Integer values are normally used here. Note that the second column of “Weight” describes the percentage of that response area’s total credit a student receives for selecting that particular answer. Real values are normally used here and they may add up to more than one. The final score that a question designer question receives is a percentage and the total number of points possible will be assigned when a user creates an assignment.

Note that the exact text match presents no problems unless special characters are involved. It is really the browser that is sending information on the particular choice a user has made in the pull down menu to MapleTA for grading and there are different ways of representing special characters that will vary between browsers. Thus, you are strongly cautioned to never place any special characters in a pull-down menu – no greek letters,
degree symbols, etc. – stick with letters and numbers. In addition, any special formatting like superscript, subscript, strong, emphasis, underlining, etc. will not work either.

**Example 11.1 – A Question Designer Narrative on the Constituents of Matter**

As an example of using this tool and its capabilities, let’s write a short reading quiz on the basic constituents of matter and their properties that might occur in a review lecture in an introductory physics, chemistry, or astronomy course. We will display a variety of different skills useful in building such questions in sections and then assemble the final product.

We can start out testing students on the location of the particles – that protons and neutrons are in the nucleus. However, we would like to ask it in a randomized way – we would like to give them one of the two that are in the nucleus and have them identify the other – and we would like this to randomize each time. Note that our algorithm correctly handles capitalization issues.

![Image of the question and algorithm variables](image-url)

*Figure 111-2 – The Constituents of Matter Question and Algorithm Variables*
Next let's test students over the nomenclature used to describe the number of particles in atoms. We will randomize between six different nuclei (but a larger number could easily be used) and ask students about the number of protons, neutrons, and electrons in an atom.
Certainly more and better distractors could be added to these pull-down menus. Authors will need to decide for themselves how rigorous they want these type of assignments to be.

Let’s create a third question sub-part exploring student understanding regarding the number of constituents. The correct answer of this question will vary depending upon randomization. Since sometimes one answer is correct and another time a different answer is correct, they both must be placed by randomization into the appropriate “correct weight” slot and at other times into the appropriate “incorrect weight” slot. In addition, there is a third answer that is always wrong and doesn’t require randomization. More than two correct answers requires more complicated “switch” randomization, but the principles are the same.
Combining all 3 subparts of the question, yields the following version shown below.
Example 11.2 – A Question Designer Narrative Section on the Nature of Light

As one final example involving randomization that occurs when using the List response area, let’s look at a section of a reading quiz over the wavelength and frequency of light. This construction involves the randomization of two criteria simultaneously and requires nested switch statements.

Figure 111-9 – Light Wavelength/Frequency Question

Figure 111-10 – Nuclei Nomenclature List Response Area and Algorithm Variables
12 Creating Questions using Tables with Randomized Entries

Sometimes a story problem is not the best way to provide data for a problem and a picture does not adequately randomize the material. Tables of data are everywhere and it should be no exception that they’re used in MapleTA questions. We will use our previous knowledge regarding using random numbers to create a problem which utilizes a table as the primary student resource used to answer the question. Using a table requires the student to parse information to determine what data to use, so we will also provide a variety of extraneous information in our table that might be used – or might not – depending on what question we randomly ask in the question.

After completing this tutorial you should…

- Be able to use HTML in the question and algorithm field to create a table in a MapleTA question.

New Functions

No new MapleTA functions are introduced, but this tutorial relies heavily on table construction tags in HTML. HTML is the markup language used to display information in a web browser and except for the incorporation of MapleTA variables this information is not specific to MapleTA. This section will review the HTML tags and properties used:

- `<table>…</table>` - contains a table which contains multiple table rows (`<tr>`). We will use the properties “cellpadding” and “border” to manipulate the appearance of the table.

- `<tr>…</tr>` contains a table row which contains multiple table cells (`<td>` and `<th>`).

- `<td>…</td>` - contains the data displayed in a cell. We will be using the properties “colspan” which indicates how many columns this cell with take up and “rowspan” which describes how many rows a cell will take up.

- `<th>…</th>` - a special type of cell declaration that indicates a header for a table (the text in the cell is automatically centered and bolded)

Please reference [http://www.w3schools.com](http://www.w3schools.com) and the sections on Learn HTML/HTML tables for additional information.
Tables in a Web Browser

Note that the MapleTA question designer has a crude WYSIWYG for table creation. When you click on the ‘Source’ button in the upper left of the Question Text editor’s toolbar, you will see that the question uses HTML to format the question. When we use a variable in MapleTA this is where the contents of the variable are really inserted (in that appropriate section of the source). So we can use HTML in our algorithm defined variables to pass along formatting to the question. We are generally not so worried about passing along bold and italic type, but getting a table formatted properly can add to the question.

A basic 2x2 table that labeled the quadrant of its cells would consist of the following HTML:

```html
<table>
  <tr>
    <td>Upper Left!</td>
    <td>Upper Right!</td>
  </tr>
  <tr>
    <td>Lower Left!</td>
    <td>Lower Right!</td>
  </tr>
</table>
```

Figure 122-1 – A 2 x 2 Table Labeling Quadrants

Example 12.1 – A Grocery List Table with Randomized Elements

We will construct a simple accounting problem using a table and then grade a student’s ability to calculate a total. Such questions are best constructed as Question Designer Questions so that they may be expanded in the future.

### Grocery List

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost per Item ($)</th>
<th>Number of Items</th>
<th>Total Cost for Item ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Milk</td>
<td>3.29</td>
<td>3</td>
<td>9.87</td>
</tr>
<tr>
<td>A2</td>
<td>B2</td>
<td>C2</td>
<td>D2</td>
</tr>
</tbody>
</table>

Where the contents of the following “cells” are specified in the algorithm field:

- A2 could simply be a randomized string: $\text{item2}=\text{switch (rint(2),"bread (loaf)","butter (lb)");}$
- B2 could simply randomize a cost and format it as money: $\text{cost2=\text{numfmt}("#.00", \text{rand}(1,2,3));}$
- C2 could simply randomize an integer: $\text{num2=\text{range (2,5);}}$
- and D2 should be a response area specified as numeric with a percentage error and no units: $\text{totalcost2=\text{numfmt}("#.00", \text{cost2*\text{num2});}}$
Note that this is the method most faculty are likely to employ to create MapleTA questions with tables: create your table in HTML and then insert MapleTA randomization commands and student response areas. Using these techniques much larger more complicated “spreadsheet-like” questions can be constructed. You are encouraged to become proficient with making questions with tables using these methods before attempting the more advanced techniques (encapsulating table components as string) that follow.

**Example 12.2 Assembling a Table With Random Elements by String Concatenation**

Using the HTML structure of a table, we will assign variables with the table elements in them. This table will be a fixed size, but will have randomized elements in the cells. We will create each row with its own variable then concatenate the rows together to save our final table in a single variable. This will all be done using the algorithm field.

The string concatenate strcat() command is used to assemble each row. Note the structure of the table in the simple quadrant example above. Each row has opening and closing tags (<tr>..</tr>), and cell tags to encapsulate the elements in each row (<td>..</td>). So if we want a row that displays 3 pieces of information, at the minimum we need the following tags: <tr><td>Element 1</td><td>Element 2</td><td>Element 3</td></tr> (at the end we will add the <table>..</table> tags as we’re working inside-out). We will replace the “Element #” text with our variables. In this example, we are working with a small table of costs and quantities. The question will ask the student to find the total price of one of the products (Apples or Pears).
All of the numbers in the example above are randomized. The quantity is an integer from 3-9, and the price varies between $1.00-2.00. We use a randomized variable to choose between Apples and Pears for the question, and the answer is the appropriate randomized quantity multiplied with the randomized unit price. For the first row we now have:

```maple
$row1 = strcat("<tr><td>","$text1","</td><td>","$quant1","</td><td>","$price1","</td></tr>");
```

It is important that we keep the data in each column aligned, so it is up to you to keep the number of elements in each row consistent (sometimes a little bit of trial and error is necessary). We have just inserted our variables between the cell tags and that completes our row. To put it all together, we will use `strcat()` again to add our header row and the outer table tags.

```maple
$table = strcat("<table><tr><th>Item</th><th>Quantity</th><th>UnitPrice</th></tr>","$row1","$row2","</table>");
```

While it may not be obvious right now why we choose to build tables this way in MapleTA, this method gives us the tools to do more powerful things with tables later. We could easily just create a large table string with all of the sub-components inside it, or even simpler yet we could just add all of the variables of the randomized components to a table using the WYSIWYG question editor. Creating the sub-components for a table and then assembling them will help us create a table with variable numbers of rows and/or columns in the next tutorial.

The algorithm, in its entirety:

```maple
$num = rint(2);
$text1 = "Apples";
$text2 = "Pears";
$numtext = switch($num,"$text1","$text2");
$price1 = numfmt("#.00",rand(1,2,3));
$price2 = numfmt("#.00",rand(1,2,3));
$quant1 = range(3,9);
$quant2 = range(3,9);
$row1 = strcat("<tr><td>","$text1","</td><td>","$quant1","</td><td>","$price1","</td></tr>");
$row2 = strcat("<tr><td>","$text2","</td><td>","$quant2","</td><td>","$price2","</td></tr>");
$table = strcat("<table><tr><th>Item</th><th>Quantity</th><th>UnitPrice</th></tr>","$row1","$row2","</table>");
$answer = switch($num,$price1*$quant1,$price2*$quant2);
```

We will use the variables $numtext and $table in the question text to give the student data to work with and $answer is our numerical answer.

MapleTA Randomization Manual 60
Example 12.3 Adding Feedback with Strings

The next step is to add feedback to our question. Our feedback table will have an added column to indicate the sub-total for each row. The most straightforward method will be to replicate the table we created for the example and just add an extra column. However, with a little bit of planning we can add our feedback text efficiently and not replicate the entire table. Replicating the table may not seem difficult for the small example, but if a larger, more complex table is used then the method described below will be invaluable.

We will assemble the feedback table in $fbtable, but first we need to change our $row# variables. We will remove the ending row tag (</tr>) from the $row# strings which will allow us to add a dynamic amount of columns (which is necessary to add our sub-total column for feedback). Our $row# variables will now only have the beginning row tag (<tr>) and each of the ‘original’ table elements (the <td>..</td>). In the original table, we will just add an ending row tag to the final $table declaration to complete the row. That will not affect how the table is displayed. This method of leaving the $row# variables open ended allows an addition of an extra column for our feedback. $fbtable will look very similar to our main table, but between the $row# and ending tag we will add our new column of information. Examples of the changes are in the code boxes below.

```plaintext
$row1 = strcat("<tr><td>","$text1","</td><td>","$quant1","</td><td>","$price1","</td>;");
```
To further facilitate the new column, we also created new variables $subtot\# (since we cannot include the formula in the strcat() function). The $subtot\# variables are inserted into the feedback table and saved us from having to rewrite all of the information. The next chapter will use this technique to create a more complex table with variable number of rows.

**Extensions**

- Add 2 cells to each row. One which indicates the units for the quantity and one which indicates the units for the cost. (eg. lbs and dollars per lb)
13 Randomized Table Configurations

Tables can be used for more than just simple shopping lists. In this tutorial we will be generating a table with a variable number of rows. This enables the table to look different to each student and prevent “pattern matching”. The randomized table size also helps to emulate real life scenarios where not every spreadsheet and table will be simple and easy to read to extract the necessary data. Most of the previous tutorials have dealt with questions where the student is given only the information necessary for the question. Here the student is given more data than is necessary to solve the problem. The techniques displayed in this chapter are complicated and time consuming to implement and should only be used in questions which will be used repeatedly by students.

After Completing This Tutorial You Should…

- Be able to assemble variable size tables
- Be able to manage extra or leading data in the algorithm field

New Functions

No new functions are used in this tutorial. However, this question relies heavily on using HTML to construct tables which is covered in the previous tutorial.

Ex. 13-1 Randomized Table Rows

Let’s quickly show how randomization can be used to restructure a table. Note that I have simply added the variable $middleRow to the table example from the previous tutorial.

```html
<table>
<tr>
<td>Upper Left!</td>
<td>Upper Right!</td>
</tr>
middleRow
<tr>
<td>Lower Left!</td>
<td>Lower Right!</td>
</tr>
</table>
```

In the algorithm section, I would define $middleRow as follows:

```plaintext
$num1=rint(2);
$middleRow = switch($num1,"","<tr><td>Lower Left!</td><td>Lower Right!</td></tr>" );
```

Thus, this middle row will be present in our table half of the time depending upon the value of $num1. It is possible to place variables and student response cells within these randomly appearing table sections. However, it is wise to begin by simply placing distractors in randomly appearing tables.
Ex. 13-2 Randomized Tables and Table Contents

The variables we will use in this question aren’t much different than any of our other randomized questions; we’re just using more of them. In this question, we are asking students to find the variance given a forecast.

![Question Name: Tutorial - Randomized Tables and Numbers](image)

Use the forecasting chart below to answer the following questions:

<table>
<thead>
<tr>
<th>Task</th>
<th>Quantity</th>
<th>Units</th>
<th>Unit Price</th>
<th>Quantity</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>570</td>
<td>SF</td>
<td>$850</td>
<td>635</td>
<td>$900</td>
</tr>
<tr>
<td>B</td>
<td>15</td>
<td>SY</td>
<td>$1800</td>
<td>65</td>
<td>$1325</td>
</tr>
<tr>
<td>C</td>
<td>25</td>
<td>TN</td>
<td>$4800</td>
<td>20</td>
<td>$4400</td>
</tr>
</tbody>
</table>

Using the table, find the variance for task C.

Normally this chart would include labor costs (and usage) separate from material costs, but for simplicity in this example – we are just using a singular unit price (presumably of labor and material costs combined). To vary the problem, we will create a chart with 2-5 different cost-tasks. Each line has its own range that the budgeted quantity and prices vary in, and the actual costs and unit consumption are based on the budgeted value. After assembling our table, and storing all of the related values, we will identify a row we will ask students about and evaluate their numeric answer.

The variables used in the first section are:

$num$ – the number of rows we are going to display

$num#$ - (where # is a number 1-5) the number of budgeted units in line #, this is set to zero if the line is not used so we do not add this to any totals later on

$num#a$ – the number of actual units in line # (note – this is determined by using some variations of $num#$), this is set to zero if the line is not used so we do not add this to any totals later on

$price#$ - the budgeted per-unit cost of line #

$price#a$ – the actual per-unit cost of line #

$row#$ - the HTML used corresponding to row #

$qtable$ – the final output variable which assembles our table
Most of the first 21 lines should look familiar. For the sake of consistency, we could have also used the numfmt() option to cast all of our numbers as money when displayed, however using \$ to insert a dollar sign for each value is just as useful here (and probably neater overall). Now, let’s focus on the last 8 rows in figure 13-1. The first thing to notice is that each of the $row# variables are the HTML starting their appropriate row. There are no closing table row tags because we want to reuse these strings for our feedback variable later on (by adding more cells in the row with row-specific variance results). The next thing to take note of is that the variables for row 3-5 are all being checked with an if-statement. The string is blank if the row is unused because this will help us when assembling the whole table. We could put the if statement in the table variable, but since these strings are being used for feedback as well – this is the most efficient usage.

The variable $qtable is where we are storing a master string which we will insert into the question text. This is an entire table, assembled and ready to display. The first two lines of the $qtable declaration is the header for the table, and each of the $row# strings are concatenated at the end. Note that our unused rows are empty, and the left over closing row tags will not affect the code. For completeness, they could be manipulated with an if-statement (like we did the 3-5 $row# variables), but this works so there isn’t a need to muddy an already complex algorithm field.

So we have built our table. However, we have not actually asked a question or created our feedback to the student. Our feedback will include a chart that has the appropriate subtotals and grand totals added to the chart (this is where our previous $row# without the end row tags will come in handy). This is shown in the second half of our algorithm field.
The variables used in this section are:

- \$var# - the computed variance for row #
- \$numa – the row which we will ask our question about (0 being the entire job)
- \$qtext – assembled text for our question
- \$answer – the numeric answer
- \$vartotal – the sum of all the variances for display in the feedback chart
- \$answera – the answer to a secondary question
- \$feedbacktext – our feedback assembled with the answers and displayed to the student

The last few lines of our algorithm are all relating to the answers to the problem. We have determined all of the values in the table previously, now we are going to use those values to ask a varying question of the student. Here is where \$numa comes in. This will determine what row we ask the student about. If \$numa comes out to zero, then we ask about the total variance. \$answer is the numeric response expected from the student. One thing special about the total \$var# values: since we made the unit variables 0 if the row was not included, then these variances will always be 0 so no additional gating is required.

We also determine if the variance in our \$answera is negative or positive to ask a corollary question regarding the type of forecast is associated with that row. So \$answera is the value which will be given to a multiple choice question (with a possible value of 1 or 2).
Figure 13-4 - Table Randomization Question Text

See the previous tutorials for details on how to insert the answers in the question designer. $answer is the numeric answer and $answera is the multiple choice answer. We also included $feedbacktext into the feedback field with a brief description about the problem.

Extensions

- Add an additional row (you’ll need to create variables: $num6, $num6a, $price6, price6a, $row6, and $var6 and edit every other non-numbered variable except $numa and $answera)
- Add a subtotal row to make the ‘total variance’ question slightly quicker
- Expand the problem to include labor as a separate category for each task
14 Case Study – Randomized Planetary Positions

This chapter will lead the reader through the steps that were used in creating a set of a randomized exam questions for introductory astronomy. The goal here was to create a substantial exam question that could be used as part of a public exam bank – so it needed to be very well randomized to prevent pattern matching by students. This particular question relies on a substantial set of diagrams more strongly than MapleTA commands and we view this approach as an effective way of creating a large body of content. The goal is to give users a good feeling for what is possible in creating randomized assignments, but developing the experience to create a question like this one is a long journey.

This question is part of a learning cycle consisting of:

- A preclass lecture in Adobe Presenter explaining the relevant concepts (and simplifications) used in this exercise
- A preclass quiz over the concepts
- An inclass activity over these same concepts, but at a slightly higher level than is shown here
- A public practice exam where students would get one of 5 (easily increased) randomized, but quite similar versions of this assignment. Students are allowed unlimited practice on this assignment.

The core of this question are diagrams created in a simple graphics program (Microsoft Word or Powerpoint) that are named quite similarly:

http://cse.unl.edu/~klee/mapleta/ast103/Segment2Graphics/planpos/picX.gif

where X is a number between 1 and 5 (there are plans for additional possibilities). The planetary positions (and thus the correct answers) are different in each version of the question. The MapleTA code then randomly selects one of the 5 images and “knows” the matching set of correct answers and distractors. Using a “randomized graphics bank” is very solid approach to creating a thoroughly randomized question that will get considerable reuse.

This assignment makes use of .qu codes for creating the question. In general, .qu codes do not work well enough to make them a recommended way of designing questions. However, this question is an exception – a question where the code is fairly simple and the number of randomization possibilities is quite large – which would be hard to make in the MapleTA online editor. It makes more sense to construct the code in a text editor and upload it into MapleTA. Another situation where the .qu codes format is useful is when you are moving a large number of questions from another brand of testbank into MapleTA and and need to do a lot of cutting and pasting.

The worksheet that would be completed in class follows in Appendix A. Note that actually having students draw positions on a horizon diagram works well for an in class exercise, but would not work in MapleTA. Thus we have moved to a simple table that allows students to select a text phrase specifying the location of the planet.
The first step is to take the graphic in the worksheets and create alternate versions of it in Microsoft Word or a graphics programs where the planets are at other locations.

A few simple rules exist for creating questions in .qu codes:

- Make a simple example in the online editor and look at the syntax of the .qu codes. You can usually see how to randomize the code.
- Make sure that the number of factors in the weighting line exactly match the number of students resource cells.

Note that the question code is quite straightforward:

```
qu.1.1.question=
<h4><center>Planetary Positions in Horizon System</center></h4>
<p>A figure is available on <a href="$url" target="_blank">this link</a> showing positions for the sun, Earth, inner planets (on the blown-up top section), and outer planets (on the bottom section). Note that the perspective is looking down onto the solar system from a point near the North Celestial Pole and the sizes of the bodies and their orbits are not to scale. Complete the table below describing the positions of the planets.</p>
<br />
<table border="1" cellspacing="5">
<tr><th>&nbsp;</th><th>Location in sky at sunset</th><th>Location in sky at midnight</th><th>Location in sky at sunrise</th></tr>
<tr><td>Mercury</td><td>$1</td><td>$2</td><td>$3</td></tr>
<tr><td>Venus</td><td>$4</td><td>$5</td><td>$6</td></tr>
<tr><td>Mars</td><td>$7</td><td>$8</td><td>$9</td></tr>
<tr><td>Jupiter</td><td>$10</td><td>$11</td><td>$12</td></tr>
<tr><td>Saturn</td><td>$13</td><td>$14</td><td>$15</td></tr>
</table>

The algorithm code is also straightforward:

```
$num1=rint(5);
$Ansa="Planet will be low over the E horizon.";
$Ansb="Planet will be somewhere (high) up in the sky.";
$Ansc="Planet will be low over the W horizon.";
$Ansd="Planet will not be visible anywhere in the sky.";
$path ="http://cse.unl.edu/~klee/mapleta/ast103/Segment2Graphics/planpos/pic";
$num2=$num1 + 1;
$url=strcat($path,$num2,".gif");
```
The only remaining piece of code is the randomized version the list pull-down menu, which also must go in the algorithm field.

$\text{\$Ans1a=switch($num1,"$Ansc","$Ansa","$Ansc","$Ansd","$Ansc");}$
$\text{\$Ans1b=switch($num1,"$Ansd","$Ansb","$Ansd","$Ansa","$Ansd");}$
$\text{\$Ans1c=switch($num1,"$Ansa","$Ansc","$Ansa","$Ansb","$Ansa");}$
$\text{\$Ans1d=switch($num1,"$Ansb","$Ansd","$Ansb","$Ansc","$Ansb");}$

Each of the 15 numbered blanks in the table in the question needs a block of code like that shown below. (Note that you cannot put algorithmic code in the “credit” fields of a MapleTA question. It would be easier to code if it was possible. Evidently the software author thought there was too much potential for error to allow that.) The correct answer must go in a field 1 all of the time. Thus, we need to randomize the correct answers and the wrong answers.

\text{\texttt{qu.1.1.part.1.mode=List@}}
\text{\texttt{qu.1.1.part.1.display=menu@}}
\text{\texttt{qu.1.1.part.1.display.permute=true@}}
\text{\texttt{qu.1.1.part.1.grader=exact@}}
\text{\texttt{qu.1.1.part.1.credit.1=1.0@}}
\text{\texttt{qu.1.1.part.1.credit.2=0.0@}}
\text{\texttt{qu.1.1.part.1.credit.3=0.0@}}
\text{\texttt{qu.1.1.part.1.credit.4=0.0@}}
\text{\texttt{qu.1.1.part.1.answer.1=\$Ans1a@}}
\text{\texttt{qu.1.1.part.1.answer.2=\$Ans1b@}}
\text{\texttt{qu.1.1.part.1.answer.3=\$Ans1c@}}
\text{\texttt{qu.1.1.part.1.answer.4=\$Ans1d@}}$

If you would appreciate the complete code to this example, please contact the MapleTA Support Desk.
Chapter 14: Appendix A – In Class Worksheet: Planet Locations

**Directions**: The figure on the following page shows positions for the sun, Earth, inner planets (on the blown-up top section), and outer planets (on the bottom section). Note that the perspective is looking down onto the solar system from a point near the North Celestial Pole and the sizes of the bodies and their orbits are not to scale.

You should first add an arrow indicating the direction of rotation of Earth and then add the times for noon, 6 pm (sunset), midnight, and 6 am (sunrise) to the diagram. You are encouraged to use a ruler to create horizon planes at the appropriate longitudes on Earth to allow you to estimate the locations in the sky (for Lincoln) for each of the planet/time combinations given below.

1. Add the letters corresponding to the following planets at the given times to the horizon diagram below if the planet is visible. You may assume that the planets are on the celestial equator (CE).
2. If the planet is not visible – estimate and indicate the length of time until it rises.

The first planet and time is done for you.

A) Mercury at sunset  
B) Venus at sunset  
C) Mars at sunset  
D) Jupiter at sunset  
E) Saturn at sunset  
F) Mars at midnight  
G) Saturn at midnight  
H) Mars at sunrise  
I) Jupiter at sunrise  
J) Saturn at sunrise

3. This exercise instructs you to use the simplifying assumption that all planets are on the celestial equator. Lightly shade the region of the horizon diagram where a planet can possibly be located over a year’s time. Then use this region to verbally describe where a planet can never be for our latitude.
Chapter 12: Appendix B – Graphic Example

<table>
<thead>
<tr>
<th>Planet</th>
<th>Sunset</th>
<th>Mid</th>
<th>Sunrise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mer</td>
<td>D</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>Ven</td>
<td>D</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>Mars</td>
<td>C</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Jup</td>
<td>A</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>Sat</td>
<td>D</td>
<td>D</td>
<td>A</td>
</tr>
</tbody>
</table>
Description: Planetary Positions in Horizon System

Jump To: Question | Information Fields

**Question:**

Planetary Positions in Horizon System

A figure is available on [this link](https://maple.ulei.edu/mapleto/contentmanager/DisplayQuestion.do) showing positions for the sun, Earth, inner planets (on the blown-up top section), and outer planets (on the bottom section). Note that the perspective is looking down onto the solar system from a point near the North Celestial Pole and the sizes of the bodies and their orbits are not to scale. Complete the table below describing the positions of the planets.

<table>
<thead>
<tr>
<th></th>
<th>Location in sky at sunset</th>
<th>Location in sky at midnight</th>
<th>Location in sky at sunrise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mercury</td>
<td>Planet will be somewhere (high) up in the sky.</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
</tr>
<tr>
<td>Venus</td>
<td>Planet will not be visible anywhere in the sky.</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
</tr>
<tr>
<td>Mars</td>
<td>Planet will be low over the E horizon.</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
</tr>
<tr>
<td>Jupiter</td>
<td>Planet will be low over the W horizon.</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
</tr>
<tr>
<td>Saturn</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
<td>[Click For List]</td>
</tr>
</tbody>
</table>