CS 105 – Introduction to Scientific Computing

Assignment 6 – Branching Scripts

Objectives

After completing this assignment you should be able to:

1. Use if/elseif/else statements to perform branching in your scripts
2. Write Boolean expression to determine which branches to execute

Overview

Thus far all of our programs have run every single command exactly once. But many programs only want to run certain commands in certain cases. Others want to choose which set of commands to run based on some condition.

In this assignment you will practice write scripts that involve branching statements and writing Boolean expressions to determine when to run different blocks of code.

Part I: Textbook Problems, Section 3.8 (page 143)

#3.1 a,b,f

#3.4

#3.11

Part II: Computing Braking Distance

The formula for braking distance is

$$d = \frac{v^2}{2g(f+G)}$$

where *v* is the velocity, *f* is the coefficient of friction, and *g* is acceleration due to gravity (32.2ft/sec^2), and G is the roadway grade as a percentage.

You want to compute the breaking distance and *ensure* valid inputs. Here are some cases to consider:

1. If (f+G) is negative, then d will be negative (this doesn't make sense)
2. If (f+G) is zero, then the formula will divide by zero
3. Velocity should not be negative
4. Friction should be in range of [0.1, 0.9]
5. Grade should be in the range [-1, 1]
6. All inputs should be numeric

When a program accepts input from a user, the programmer should be aware that the input could be *anything* and needs to safeguard against bad cases.

Program Flow

You must invent the algorithm to complete this task. However, this is the general pattern you can use repeatedly to test each input:

1. Get the user inputs as a string: input(prompt,'s');
2. Call the function str2num to try to convert the string into a number. str2num returns an empty vector if the argument string doesn't consist of numerical characters
3. Call function isempty to determine if the vector returned by str2num is empty. If its empty then the input wasn't a number so display an error message and quit.
4. If the value is numeric (passed step #3), then make sure it's in valid range. If it isn't, return an error message and quit.

Test Plan

Be sure to test every type of incorrect input:

- Non-numerical velocity
- Non-numerical friction
- Non-numerical grade
- Negative velocity
- Coefficient of friction less than 0.1 or greater than 0.9
- Road grade less than -1 or greater than 1
- Sum of friction and road grade less than or equal to zero.

Example Input/Output Behavior

```
Initial velocty in ft/sec: -2
Please enter a positive velocity.
-----
Initial velocty in ft/sec: abc
Input abc is not a number. Please enter a number.
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: -2
Coefficient of friction must be in the range [0.1, 0.9]
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: abc
Input abc is not a number. Please enter a number.
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: .4
Roadway grade: -3
Road grade must be in the range [-1, 1]
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: .4
Roadway grade: abc
Input abc is not a number. Please enter a number.
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: .4
Roadway grade: 0
Stopping distance = 33.4217 feet
-----
Initial velocty in ft/sec: 29.33
Coefficient of friction: .4
Roadway grade: -.5
Sum of coefficient of friction and road grade must be positive. It is
-0.1
```

Submission

Submit a z*ip* file that consists of:

1. Your PDF report that contains
    a. Your answers for Part I
    b. An explanation of what you did in Part II (what you did, how you did it, any additional things you tried or discoveries you made)
2. The .m files created to do Part II