Assignment #4

Introduction to C Programming – COP 3223

Objectives

1. To reinforce basic programming constructs
2. To learn how to use for loops

Introduction: Mission to Mars

Your friend has been playing a new Mars Colony simulator nonstop! They are always talking about how cool it would be if they could be a on the first real-life mission to Mars! To amuse your friend, you have decided to create a series of programs about the possible first colony on Mars.

Problem: How Do We Land The Shuttle? (marslanding.c)

Once we have all of our fuel and equipment purchased and packed, we will be ready to leave! Our next problem will be landing our shuttle on Mars. Given the conditions on the red planet, this can be very difficult. To practice, we have created a landing simulator based on the Mars Rover program.

To begin this program, you should first ask the user how many days they trained on the simulation. Then, for each day ask how many simulations will be run. For each simulation, ask the user how many adjustments were needed for landing.

If fewer than 5 adjustments are made, that simulation is considered a success. More than or equal to 5 adjustments indicates the simulation is a failure. Track the number of failures for each day and print it so the user can see how their astronauts are doing.

Input Specification

1. The number of days will be a positive integer.
2. The number of simulations will be a positive integer.
3. The number of adjustments for each simulation will be an integer greater than or equal to 0.

Output Specification

Output the number of successful simulations for each day.

Day #X: There were Y successful simulations.

Output Sample

Below are some sample outputs of running the program. **Note that these samples are NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run

```
How many days did we use the simulator?
3

How many simulations did we run on day #1?
2

How many adjustments were needed for run #1?
10
How many adjustments were needed for run #2?
7

Day #1: There were 0 successful simulations.

How many simulations did we run on day #2?
4

How many adjustments were needed for run #1?
7
How many adjustments were needed for run #2?
4
How many adjustments were needed for run #3?
5
How many adjustments were needed for run #4?
3

Day #2: There were 2 successful simulations.

How many simulations did we run on day #3?
3

How many adjustments were needed for run #1?
4
How many adjustments were needed for run #2?
4
How many adjustments were needed for run #3?
3

Day #3: There were 3 successful simulations.
```

Acceptable Resources

Remember, the use of online help sites is strictly prohibited. The only acceptable resources for these assignments are below:

- Course Webcourse
    - In particular: Week 5 – Loops

- Course Textbook
    - Programming Knights: An Introduction to Programming in Python and C by Arup Guha
- Professor Guha's Course Archive
    - http://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3223/
- Course TAs and Instructor Office Hours
    - Getting Help: https://webcourses.ucf.edu/courses/1336411/pages/getting-help

Style Notes

Please review the course Style Guide on the webcourse, with special attention to the following notes:

- comment major sections of code addressing: "What does this block do?" and "Why did I implement this block in this way?"
- place comments above the line(s) to which it applies
- use inline comments (//) and leave one space between // and the comment's first character
- All variables should be declared at the top of your functions (in this program, only main is needed) and should have meaningful names
- Be sure to declare `main` with: int main(void) {
- Indent the contents of main four spaces or one tab
- leave a space on both sides of any binary operators you use in your code (i.e., operators that take two operands). For example, use (a + b) - c instead of (a+b)-c.
- keywords if, while, and for should have a single space after them
- contents of if statements and loops should be indented four spaces or one tab
- conditions should not have any space immediately after each `(` or immediately before each `)`.

Deliverables

One source file – *marslanding.c* – is to be submitted over WebCourses.

Restrictions

Although you may use other compilers, your program must compile and run using a standard C Development Environment. Your program should include a header comment with the following information: your name, assignment number or title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.