Assignment 3 Description

Conditionals

Due date: Wednesday February 10th 2016 6:00 pm

Notes:

- **This assignment is individual work**. You may discuss the questions and problems with anyone, but the work you hand in for this assignment should be your own work.
- **Each question indicates what to hand in.** Often, it will be a single document for **each question**. You must give your document the name we prescribe for each question, usually in the form **aNqM,** meaning Assignment N, Question M. You should make sure your name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you. Do not submit folders, zip documents, even if you think it will help.
- **Hand in your work using Moodle.**
- **Moodle will not allow you to submit work after the due date.** It is advisable to hand in each answer that you are happy with, as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Questions are annotated use descriptors like "easy" "moderate" and "tricky". All students should be able to obtain perfect grades on "easy" problems. Most students should obtain perfect grades on "moderate" problems. The problems marked "tricky" may require significantly more time, and only the top students should expect to get perfect grades on these. We use these annotations to help students be aware of the differences, and also to help students allocate their time wisely. Partial credit will be given as appropriate, so hand in all attempts.
- **How to find your C++ programs to submit to Moodle for grading.**
    - When you hand in your C++ programs, hand in the .cpp file only, not the whole Eclipse project.
    - To find your .cpp files that you create as part of your projects, open up Explorer/Finder/Dolphin and navigate to the folder/directory that you selected as your 'workspace.' Within that folder you'll see folders for each of the projects that you created. Navigate into the appropriate one, and then into the "src" folder; you'll see your .cpp file there.

Question 1 (6 marks)

Purpose: Practice simple conditionals, and to practice testing your program, and submitting evidence of testing.

Degree of Difficulty: Easy

Write a program that will ask the user for a **positive** integer and then display on the console whether the integer is divisible by 7 (without a remainder). For example, when you run your program, the following interaction would be appropriate:

```
Please enter an integer: 29

That is not divisible by 7.
```

(The green text represents the data typed by the user in the console.)

Of course, your message would be different if the user entered a number that **is** divisible by 7:

```
Please enter an integer: 308

Yeah! 308 is divisible by 7!
```

Your program only has to ask for one number at a time, but you should run your program several times, testing with numbers that you know are divisible by 7, and not divisible by 7.

Note: Your program should ensure that the number entered by the user is a positive number, and if not display an error message and stop. The following interaction would be appropriate:

```
Please enter an integer: -40

That wasn't a positive number!  Goodbye!
```

What to hand in:

1. Your C++ code in a file called **a3q1.cpp**. Be sure to include your name, NSID, Student Number, lecture section and tutorial section in comments at the top of your file.
2. Some examples of your testing, in a text file called **a3q1.txt** (.rtf, and .doc are also okay).

Evaluation:

- 0/6: Nothing submitted, or files cannot be opened.
- -1 marks: for lack of identification in the files (Name, NSID, Student number, Tutorial section, and lecture section).

- 3 marks: Your program correctly uses a conditional to identify if the number is divisible by 7. Marks will be deducted for incorrect use of if-else, or incorrect conditions, and for poor indentation and presentation of the program.
- 2 marks for testing. You should show the results of several runs, showing that your program works correctly.
- 1 mark: You validated and ensured that the number entered by the user is a positive number.

Question 2 (12 marks)

Purpose: To practice using conditionals in C++ programs; to practice designing effective test cases

Degree of difficulty: Moderate.

Ernie and Bert are arguing about which of them is older, and they need your help to figure it out. Write a C++ program that asks the user for the birthdays of both Ernie and Bert, and responds by indicating which person is older (or if they are the same age). Here's a demonstration, copy/pasted from the Console window.

```
Assignment 3 Question 2: Birthdays
Please enter Ernie's birthday, using YYYY MM DD: 1984 6 16
Please enter Bert's birthday, using YYYY MM DD: 1975 11 6
Bert is older than Ernie.
```

Note that the user typed the numbers that are displayed using green colored text.

You should do some problem solving before you start. What will your algorithm be? Having a good plan before you start will make you more productive.

Your task involves the following steps:

1. Before you start designing your program, you might think that there are 3 possibilities: 2 cases in which one person is older, and 1 case that the people are the same age. However, there are actually more than 3 possible scenarios that you will have to consider in your design, and your program must answer correctly for all of them. We call such scenarios "test cases", especially if we use them to test the hypothesis that a program is correct.
   Hint: You'll need 7 test cases (examples) to cover the possibilities. If you can't imagine why you need 7 test cases, you haven't fully understood the problem.
   Note: Assume your user will type integers as requested. You do not have to worry about the user entering other kinds of data, like characters or floats.
   Note: You may assume that all data given by the user is error-free; they will not try to enter a day beyond the end of the month (ex: 2 40 [Feb 40]), a month that does not exist, a negative year, or any other kind of error like that.
2. Starting from the Hello World template in Eclipse, write a program that behaves the same way as the above example. It must ask for 2 birth dates, and must produce a statement about which person is older.
3. Show that your program gets all your test cases right. Run your program once for each test case, and copy/paste the Console window text into a document to submit for grading.

What to hand in:

1. Your program called **a3q2.cpp;**
2. A document called **a3q2.txt** (.rtf, and .doc are also okay) showing testing that you did to verify that your program works correctly (copy/paste from the console). Include your name, NSID, Student Number, lecture section and tutorial section at the top of your document.

Evaluation:

- 0/12: Nothing submitted, or file cannot be opened.
- -1 marks: for lack of identification in the files (Name, NSID, Student number, Tutorial section, and lecture section).
- 12 marks for content.
  - 7/7 if your program compiles without error, and correctly solves the problem.
  - 5/5 if your testing includes distinct 7 test cases that test the program appropriately.
  - Part marks as appropriate.

Question 3 (10 marks)

Purpose: To practice using nested conditionals and multiple user inputs.

Degree of difficulty: moderate

Let us suppose that cookies have been mysteriously disappearing from across campus. Campus Security has asked you to write them a C++ program to help them interrogate witnesses in order to catch the Great Cookie Thief.

The following facts are known about the thief:

- The thief's hat is black
- The thief's scarf is red
- The thief's fur is blue
- The thief's eyes are googly

Your program will ask the user questions, one at a time. If all of the answers given by the user match the facts above, the program will indicate that the Great Cookie Thief has been found. For example:

```
Witness, please answer the following questions.
What color was the suspect's hat?: black
What color was the suspect's scarf?: red
What color was the suspect's fur?: blue
How would you describe the suspect's eyes?: googly

IT'S HIM!!  You've found THE GREAT COOKIE THIEF!!
```

On the other hand, if at any time the user answers a question in a way that does not match the facts above, the program will **immediately stop without asking any more questions**. For example:

```
Witness, please answer the following questions.
What color was the suspect's hat?: black
What color was the suspect's scarf?: lavender

Whew!  Looks like it wasn't the Great Cookie Thief.
```

Design **five separate test cases** for your program, then run the program five times, once for each test. Copy/paste your tests from the Eclipse console window into a plain text document that you will submit for grading.

(Remember that a "test case" is a scenario which, when given to the program by user input, will provide evidence for (or against) the hypothesis that your program works correctly. It is your duty as a scientist to try to prove your program does **not** work. Only when you fail to prove it does not work should you be confident that it actually does work.)

What to hand in:

1. Your program called **a3q3.cpp;**
2. A document called **a3q3.txt** (.rtf, and .doc are also okay) showing the five test sessions that you did to verify that your program works correctly (copy/paste from the console). Include your name, NSID, Student Number, lecture section and tutorial section at the top of your document.

Evaluation:

- 0/10: Nothing submitted, or file cannot be opened.
- -1 marks: for lack of identification in the files (Name, NSID, Student number, Tutorial section, and lecture section).
- 10 marks for content.
  - 5/5: Program behaviour is correct for each test case.
  - 2/2: Test document is legible and shows all five test cases
  - 3/3: Proper indentation and presentation are used in code

Question 4 (10 marks)

Purpose: To practice checking for number ranges, practice with expressions, more practice with conditionals

Degree of difficulty: moderate, but random numbers are a new concept

Random numbers are often useful in computer programs for a variety of reasons. For this question, you will write a program that generates random numbers and then uses some conditional statements that depend on the random numbers.

First, some technical notes. To use random numbers in your program, you must include the following two lines at the very top of your program (right next to "#include <iostream>":

```
#include <cstdlib>

#include <time.h>
```

Then, right after the opening curly-brace of "int main()", you include the following line:

```
srand(time(NULL));
```

After you've done those things, any time you want a random integer, you can get one by calling the rand() library function. Of course, you will probably want to store the random number that rand() gave you in a variable. The following line of code would store a random number in the range of 1 to 100 (inclusive) in the variable myNum.

```
int myNum = rand()%100 + 1;
```

For this question, you will write a program that will generate a random number between 1 and 100 (inclusive) and then print out a message based on what the number was. Then, it will generate a second random number and print out an appropriate message again. The rules for the message are as follows:

- print out "The number was medium" if the number was greater than or equal to 25 and less than or equal to 75
- print out "The number was high" if the number was greater than 75 and less than 95
- print out "The number was low" if the number was less than 25 and greater than 5
- print out "The number was EXTREME!" if the number was less than or equal to 5, or greater than or equal to 95.

An example run of your program might look like this:

```
Welcome to the Number Generator!
```

```
The first number is: 38

The number was medium

The second number is: 3

The number was EXTREME!
```

You must then test your program until you are sure it is working. Notice that there is no console input for this program (i.e. no "cin"), so there are no test cases to design. However, because the program uses random numbers, you will have to test it several times to make sure it is working. For grading, you will hand in a log of all of your test runs (just copy-and-paste them from the Eclipse console into your testing file); the log must show that your program works in all cases (figuring out whether your log indeed shows this is part of the question!).

Finally, once you are done with that testing, remove (or comment out) the line "srand(time(NULL));" from your program, then run your program a few times and observe what happens.

What to hand in:

1. Your program called **a3q4.cpp;**
2. Your testing file called a3q4_testing.txt (.rtf, .pdf or .doc are also ok). In this file you should **FIRST** answer the following two questions (in one or two sentences each), and then include your testing results showing that your program works. The two questions are (1) How many test runs did you need to prove your program works, and why is the log you included sufficient to show that your program works?; and (2) What happens when you remove the line "srand(time(NULL));" from the program?

Evaluation:

- 0/10: Nothing submitted, or file cannot be opened.
- -1 marks: for lack of identification in the files (Name, NSID, Student number, Tutorial section, and lecture section).
- 10 marks for content.
    - 1/1: The program correctly generates two different random numbers
    - 5/5: The program prints out the right message
    - 4/4: Testing file answers the two questions and shows that the program works