University of Toronto
APS105H1F — Computer Fundamentals
Lab 2: Conditionals & Loops

Introduction

This lab is designed to give you practice with basic conditionals and loops in C++. In particular, this lab is intended to get you familiar with `if`-statements and `while` loops. You should attempt to complete the assignment *before* your scheduled lab period, as many of you will find it will take more time than will be available in the lab. Please keep in mind that the marking process will begin approximately forty minutes before the end of the lab.

A Number Guessing Game

Your assignment in this lab is to write a program that plays a number guessing game. Your program is to "randomly" pick a secret whole number in the range 0 to 100 (including both 0 and 100). It asks the user if they wish to try and guess the secret number, accepting a response of 'y' or another character. If the response is not 'y', the program exits without any further output. If the response is 'y', the program prompts the user to input numbers as guesses. For each guess, the program will inform the user whether their guess is correct, or whether the secret number is higher or lower than the user's guess. Each time the program prompts the user for a guess, it tells them the number of the guess, as the maximum number of guesses is limited to 5. If the user has not correctly guessed the secret number in the maximum allowed number of guesses, the program tells the user that they are out of guesses, and what the number was. At any time the user may enter the guess -1 to give up. If the user enters a value outside of the range 0 . . . 100, the program reminds them of the correct range and allows them to guess again without incrementing the number of guesses used. After each round of guessing, the program will ask the user if they wish to play again. You may notice that 5 guesses is not often enough to guess the right number. If you use a named constant for this value, it will be easy to modify your program to try 6 or 7 as the maximum number of guesses.

Note: You may assume the user always gives input in the correct format. For example, they will not enter `two` or `fred` when an integer value is expected.

Sample Output

To give you a more concrete sense of how your program must behave, the following is a sample session illustrating the behaviour described above. Given the same input, your program **must** generate exactly the same output (with the exception of the random nature of the number to be guessed). Failure to do so will result in marks being deducted from your grade.

```
$ ./guess
I know a number between 0 and 100 (inclusive)
Would you like to try and guess it? (I'll give you 5 tries)
Enter 'y' for Yes and any other character for 'No': y
Guess #1, Enter your guess, or -1 to give up: 50
Lower ...
Guess #2, Enter your guess, or -1 to give up: 25
Higher ...
Guess #3, Enter your guess, or -1 to give up: 37
```

```
Higher ...
Guess #4, Enter your guess, or -1 to give up: 43
Lower ...
Guess #5, Enter your guess, or -1 to give up: 41
You've run out of guesses ... the number was 38

I know a number between 0 and 100 (inclusive)
Would you like to try and guess it? I'll give you 5 tries.
Enter 'y' for Yes and any other character for 'No': y
Guess #1, Enter your guess, or -1 to give up: 134
Your guess must be between 0 and 100, please try again.
Guess #1, Enter your guess, or -1 to give up: 50
Lower ...
Guess #2, Enter your guess, or -1 to give up: 25
Higher ...
Guess #3, Enter your guess, or -1 to give up: 37
Lower ...
Guess #4, Enter your guess, or -1 to give up: 31
Lower ...
Guess #5, Enter your guess, or -1 to give up: 28
You've run out of guesses ... the number was 30

I know a number between 0 and 100 (inclusive)
Would you like to try and guess it? (I'll give you 5 tries)
Enter 'y' for Yes and any other character for 'No': y
Guess #1, Enter your guess, or -1 to give up: 50
Lower ...
Guess #2, Enter your guess, or -1 to give up: 25
Lower ...
Guess #3, Enter your guess, or -1 to give up: 12
Lower ...
Guess #4, Enter your guess, or -1 to give up: 0
Correct! You win!

I know a number between 0 and 100 (inclusive)
Would you like to try and guess it? (I'll give you 5 tries)
Enter 'y' for Yes and any other character for 'No': y
Guess #1, Enter your guess, or -1 to give up: -1
You give up? The number is 6

I know a number between 0 and 100 (inclusive)
Would you like to try and guess it? (I'll give you 5 tries)
Enter 'y' for Yes and any other character for 'No': n
```

Generating Random Numbers

In order to generate random numbers in your program, the following information should help:

1. The expression `rand() % n` will generate pseudorandom[1] numbers in the range 0 . . . (n - 1).
2. To avoid getting the same sequence of "random" numbers each time you run your program, include the statement

[1] A sequence of *pseudorandom* numbers is one that appears random, but is actually predictable, especially if the underlying algorithm is known.

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

at the beginning of your program. In order to use the `time()`, `srand()` and `rand()` functions, you will also need to place the statements

```
#include <cstdlib>
#include <ctime>
```

at the beginning of your program file.

Requirements

Your program must be in a file named `guess.c`. There must be comments at the start of the file indicating your name, student number, course code (APS105), assignment number (Lab 2), and the date you began work on the file.

In order to demonstrate your C++ knowledge, your program must also contain examples of each the following:

- named constants,
- Boolean variables,
- `char` variables, and
- *meaningful* comments (not including the comments at the start with your name, student number, *etc.*).

You must be able to compile your program and demonstrate it for the TA who marks your lab.