Week 3 of CS319: Scientific Computing (with C++)

Lab 1: Numbers and Programming

Goal: To gain familiarity with the concepts of

- basic C++ program structure;
- input and output,
- *Flow-of-control*: `if` statements, and `for` *loops*
- Computer representation of number – particularly `int`s.

You need to upload something, even if not complete, by 17:00, Friday 30 January to https://universityofgalway.instructure.com/courses/44564/assignments/139657

Lab 2 will develop this further, and will carry more marks.

Question I

Q1. (`if/else if/else`-statements) Write a C++ program that prompts the user to enter two integers, x, y, and then reports which in quadrant the point (x, y) is found, or if (x, y) is on an axis (i.e., one or both are zero.
Tip: see https://en.wikipedia.org/wiki/Quadrant_(plane_geometry) for a definition of quadrants I, II, III and IV.

Question II

Q2. (cin and while). Write a short C++ program that works as follows.

- The user is prompted for a number between 1 and 10 (inclusive), storing the input in an integer variable, n.
- A while loop is used so that, if n is not in that range, the user is prompted for it again and again, until they enter one in the correct range.
- The final value of n is displayed, along with a suitable message, that includes the number of attempts taken.

Question III

Q3. The following code snippet finds the largest `int` that is correctly representable by your computer. It also computes the time taken. (Full code at Lab1-Q3.cpp).

```
   clock_t start;
   float diff, diff_seconds;
   start=clock();

   int i=1;
   int j=i+1;
   while ( i<j )
   {
      i++;
      j=i+1;
   }
   diff = (float)(clock()-start);
   diff_seconds = diff/CLOCKS_PER_SEC;
   std::cout << "Overflow at i="<< i << std::endl;
   std::cout << "Computation took " << diff_seconds
        << " seconds."  << std::endl;
```

Question IV

Q3(a) Read the code carefully, and make sure you understand it. Test it, making sure you compile without any optimisations. Do the results agree with the theory covered in class?

Q3(b) There are other types of integers available in C++, for example, `short int`, `unsigned int` and `long int`. Try this program using `short int`s and `unsigned int`, which are stored using 2 bytes. Do you get the expected results?

Q3(c) C++ has a data type called `long int` which uses 8 bytes. Suppose you wanted to use this program to test the largest `long int` your C++ programs can represent. Estimate how long your program would take to run. **Warning: don't actually try this by running the code!!**.

Note: You can check the number of bytes that a datatype uses, with the `sizeof()` function. Given a variable, or the name of a type, it returns the number of bytes used to store it. You can use this to verify that `short int`, `int` and `long int` use 2, 4 and 8 bytes, respectively.

Question V

In Lab 2, we'll write some code that estimates the smallest and largest `float`s and `double`s that one can store, and the machine epsilon for this types.
If you'd like to get started early, think about how that can be done.