CS319: Scientific Computing (with C++)

CS319 Lab 2: `int`, `float` and `double`

02 March 2021

Goal: To study the computer representation of numbers – particularly `int`s, `float`s and `double`s.

Assignment: Submit your code as a single C++ program for Q3 and Q4. Upload it to the "Assignments... Lab 2" section of the CS319 Blackboard module.
It is **very important** that your code include comments with your name, email address, ID number, date written.
The assignment will be graded taking into account if the program compiles, if it solves Q3 and Q4, and if it includes the requested data.

Q1. The following code snippet finds the largest `int` that is correctly representable by your computer. It also computes the time taken. (Full code at Lab2-Q1.cpp [link]).

```
    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;
```

Q1(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?

Q1(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`. Do you get the expected results?

Q1(c) 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.

Q2. Write a programme to tries to compute the smallest `float` greater than zero that your computer can represent. For example, you could initialise a `float`, x, as 1.0. Then, for as long as your computer thinks that $x/2 > 0$, divide x by 2. When you are done, x should be a good approximation of the smallest number representable.
Does the answer given by your code agree with theory? If not, can you give a reason why?

Q3. Compute the smallest `float`, x, such that we can distinguish between 1 and $1 + x$. Write a C++ program to do this. (Hint: the thing you are computing is called *machine epsilon*. The Wikipedia entry for this is rather good).

Q4. Write a C++ programme that computes the *machine epsilon* for the `double` data type.