Week 4 of CS319: Scientific Computing (with C++)

Lab 2: `float` and `double`

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

Assignment: Submit your code as a single C++ program for Q4. Upload it to Canvas... `2324-CS319` ... "Assignments... Lab 2".

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 Q4, and if it includes the requested information.

Collaboration policy. Collaboration is encouraged. It is acceptable for two people to work together and submit exactly the same work. However.

- Both need to submit the code independently. (No submission, no score, no exceptions).
- Your submission must include comments with YOUR name and ID number, and also give the name of your collaborator.

Question I

Q1. 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?

Question II

Q2. Next we want to compute the largest `float` representable. This is a little trickier; whereas small floats are eventually rounded to zero (which is a number), large ones tend to infinity (which is not a number).
Try a similar approach as in Question 1, but include the `cmath` header file. Then you can use the `std::isinf()` function (for example) to test if x is finite or not. Depending on your compiler, you may have to compile against the `math` library.

Question III

Q3. (i) Compute the smallest `float`, x, such that we can distinguish between 1 and $1 + x$. Write a C++ program to do this. Some notes:

- The thing you are computing is called *machine epsilon*. The Wikipedia entry for this is rather good.
- Compilers, and CPUs, often try to be clever, and may preform interim calculations at higher precision than asked. So:
  `if ( 1.0 + x/2.0 > 1.0 )` ...
  can behave differently from
  ```
  z=1.0+x/2.0;
  if ( z > 1.0 ) ...
  ```

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

Question IV

Q4. **Assignment**: Write a single C++ program that

(i) estimates the smallest positive `double` that is representable;
(ii) estimates the largest positive `double` that is representable;
(iii) estimates the smallest positive `double`, x, such that $1 + x$ is distinguishable from 1.

When run, the programme should output your results in a suitably coherent manner.
Add comments that explain the observed output.
And don't forget to add your name and ID number (anonymous code won't be graded.)