# CS319 Lab 1: Numbers and Programming

Goal: gain familiarity with the following concepts

- Basic program structure; input and output,
- Loops (`for`)
- Flow-of-control (`if` and `switch/case`)
- Computer representation of number – particularly `int`s, `float`s and `double`s.

**Submit your answer to Q8 by 2pm, Friday, 27 January.**

Q1. Write a short programme with defines an integer `a`, and initialised its value to 9. Use the code to determine what each of the following lines of code do.

```
a=-a;       a-=a;
--a;       a=(a==a);
```

Q2. Write a C++ programme that works as follows:

- The user is prompted for an `integer n` between 1 and 19.
- If `n` is not between 1 and 19 (inclusive) and error is returned.
- Otherwise a `for()` loop is used to compute and output the first `n` powers of 3.

The output should be nicely tabulated as follows. For example, if 12 is entered, the output should look like:

```
3^1  =        3
3^2  =        9
...
3^11 =   177147
3^12 =   531441
```

The `setw`, `left` and `right` manipulators might be helpful. (*Extra: how could one set the argument to* `setw` *depending on the user input?*).

Q3. Write a program that prompts the user to enter the (pre-VAT) cost of an item, and also selects which of the following *Value Added tax* (VAT) rates should be applied: 23% (Standard), 13.5% (Reduced), 9% (Tourism), 4.8% (Livestock), and 0.0% (Zero). It should output the net price after VAT is applied.

Q4. Find out how the `switch/case` construct works. Rewrite the above example using `switch/case` instead of an `if/else` statement.

Q5. The following program snippet, which you can download from the course website as Lab1-Q5.cpp, finds the largest `int` that is correctly representable by your computer. It also computes the time taken.

```
#include <iostream>
#include <time.h>

int main(void)
{
  std::cout << std::endl <<
    "------------------------------\n" <<
    "|␣CS319␣␣Lab␣1,␣Q5,␣␣23/01/17␣|\n" <<
    "------------------------------\n";
  int i, j;
  clock_t start;
  float diff, diff_seconds;
  start=clock();

  i=1;
  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;
  return(0);
}
```

Q5(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?

Q5(b) There are other types of integers available in C++, for example, `short int`, `unsigned int` and `long int`. Try this program using `short ints` and `unsigned int`. Do you get the expected results?

Q5(c) This program takes about 9 seconds on my laptop, which has a 2.3GHz CPU. How long does it take on your computer, and how does this related to the CPU speed?

Q5(d) 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.

Q6. 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?

Q7. Next we want to compute the largest `float` representable. This is a little more tricky; where as small floats are eventually rounded to zero, large ones tend to infinity.

Try a similar approach as in Question 6, but include the header file `math.h` (and include the `h`); and use the function `isfinite` to test if $x$ is finite or not. Depending on your compiler, you may have to compile against the `math` library.

Q8. ⋆ For the remainder of this course, we won't use `float`s very much, except for certain examples. Must numerical computation is done with `double`s. 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 $\mathbf{1}$.

Add comments that explain the observed output.

**Submit your answer to Q8 by 2pm, Friday, 27 January.**

Do this by selecting the "Labs" menu on Blackboard, and uploading to the "Lab 1" section. Before you do, check the rubric to make sure you understand how your work will be graded.