QUEEN MARY, UNIVERSITY OF LONDON

MTH6150

Numerical Computing in C and C++

Exercise Sheet 8

1. Run the code on slides 6 and 7 to verify the different behaviour when passing by value or by reference.

   Also run the code on slide 14 to check that it does what it says, i.e. swap the values of the input variables. Add code to output `a` and `b` after the `swap` function to confirm this. Change the arguments in the `swap` function to passing by value

   ```
   swap(int m, int n)
   ```

   and see what happens.

2. Initialize two vectors (to contain type `double`) with values $\{2.0, 1.0, 0.0\}$ and $\{-1.0, 2.0, 0.0\}$, respectively. Calculate the inner product of these vectors using the function on slide 19.

3. Write a function `output` to output a `vector<double>` to the console (the black window that `cout` writes to), with a space between each element. The function should take a `const` reference argument.

   Write a function of the same name that outputs a `vector<int>`, also with a `const` reference argument.

   Check that these work as expected by creating each kind of `vector`, and then calling the `output` function.

4. Write two functions, called `mean` and `var`, that return the mean and variance of the input argument, which should be a `vector<double>`. So the first of these would be of the form:

   ```
   double mean(const vector<double>& v){
       ...
   }
   ```

   The mean of a sample $x_1, x_2, \ldots, x_n$ is

   $$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

   and the variance by convention is given as

   $$\hat{\sigma}^2 = \frac{1}{n-1}\sum_{k=1}^{n} (x_k - \hat{\mu})^2$$

Hint: see the function for the sum on slide 17 of lecture 8.

Also write functions `min` and `max` to find the minimum and maximum of a `vector<double>`.

Test all of these functions on a `vector<double>` containing the elements $\{1, 2, 3, 4, 5\}$. The mean and variance should be 3 and 2.5 respectively.

5. Random number generators are important in numerical applications. One of the simpler methods is called the linear congruential method. Here, starting with some initial number X_0 (which is called the seed) and constants a, c and m, with all numbers non-negative integers, we can generate a sequence

$$X_{n+1} = (aX_n + c) \bmod m$$

Here, mod is the modulus or remainder operator, which in C++ is `%`.

Obviously this generates a deterministic sequence, but for good choices of a, c and m the sequence looks like random integers between 0 and $m - 1$.

One possible choice is $m = 134456, a = 8121, c = 28411$. Write a function

```
int randlc(const int x)
```

that uses these constants and does a single update of the sequence, so the function argument is X_n and the returned value is X_{n+1}.

For some large N, e.g. 10,000, and some positive integer choice for X_0, use the function `randlc` to fill in a `vector<int>` of size N with the sequence X_1 to X_N.

Then create a `vector<double>` of size N and set the elements equal to

$$\frac{X_1}{m-1}, \ldots, \frac{X_N}{m-1}$$

These values should approximate random numbers from the uniform distribution on the interval 0 to 1. This continuous distribution has mean 1/2 and variance 1/12. Use the functions from question 4 to check the mean and variance, and also check that the minimum and maximum values are close to 0 and 1 respectively.