MTH6150

Numerical Computing in C and C++

Exercise Sheet 4

1. Try each of the following blocks of code. In each case, note down what you think the output will be before running the code, and experiment with the code, especially if you didn't correctly predict the output.

   If the program just keeps running, you can force it to stop by closing the console window (the black window that shows the output from `cout`).

   (a)
   ```
   for (int i = 1; i<10; i++) {
     cout << i << endl;
   }
   ```
   In the above example, change the `i++` to each of the following in turn:

   `i += 3`

   `i *= 2`

   Change it back to `i++`, and add an `i++;` command inside the `for` loop.

   (b)
   ```
   double S = 0;
   for (int i = 1; i<=5; i++) {
       S +=i;
   }
   cout << S << endl;
   ```

   (c)
   ```
   for(double x=0.0; x<10.0; x+= 0.5){
       cout << x << endl;
   }
   ```

   (d)
   ```
   for (int i = 10; i>=0; i--) {
       cout << i << endl;
   }
   ```
   In this example, try replacing `int` with `unsigned int`, which is a variable type that only holds non-negative integers.

   (e)
   ```
   int i = 100;
   while (i>0) {
       cout << i << endl;
       i /=2;
   }
   ```

   (f)
   ```
   int i = 100;
   do{
       cout << i << endl;
       i /=2;
   }while (i>0);
   ```

Change the initial `int i = 100;` to `int i = 0;` in this and the previous example.

(g)
```
int i = 100;
while (i>0) {
    cout << i << endl;
    if (i%6 == 0)
        break;
    i /=2;
}
```

(h)
```
for (int i = 0; i<10; i++) {
    if(i<5)
        continue;
    cout << i << endl;
}
```

(i)
```
for (int i = 0; i<10; i++) {
    if (i<5)
        continue;
    else if (i==7)
        break;
    else
        cout << i << endl;
}
```

2. Run the code on slides 11 and 12 to understand when the variables exist in the program.

3. (**Problem for handing in**) Leonard Euler discovered in 1735 that

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

Write a program to evaluate this sum for some large number of steps, which should be an input to the program, to check the formula.

Standard C++ does not know about π, so you could add the following to your code:

```
double pi = 3.14159265358979;
```

Alternatively, Visual Studio and some other compilers implement some mathematical constants. You would need to add this to the top of the code file:

```
#define _USE_MATH_DEFINES
#include <cmath>
```

and then π is then available as `M_PI`
However, this code would not be as portable, as it may not work with another compiler. `#define` is a type of pre-processor directive that we will not otherwise need in this module.

4. (**Problem for handing in**) The Fibonacci numbers are defined as follows:

$$F_0 = 0, \; F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}, \quad n \geq 2$$

Write a program that outputs all the Fibonacci numbers that are less than 1000.

One way of organising the program for the Fibonacci numbers is to declare three variables to hold the values of F_{n-2}, F_{n-1} and F_n.

Start with $F_{n-2} = 0, F_{n-1} = 1$.

Then at each iteration of the loop:
put $F_n = F_{n-1} + F_{n-2}$
put F_{n-2} equal to the current value of F_{n-1}
put F_{n-1} equal to the current value of F_n
The ratio of successive numbers tends to a limit known as the *golden ratio*

$$\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \frac{\sqrt{5}+1}{2}$$

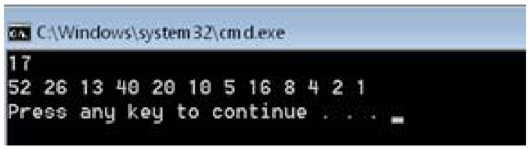
Write code to check that the ratio of the values you calculate does approach this limit for large n. Note that the square root of `x` in C++ is `sqrt(x)`.

5. (**Problem for handing in**) Define a sequence of integers $a_1, a_2, \ldots$ as follows for any positive integer a_0:

$$a_{i+1} = \begin{cases} a_i / 2 & \text{if } a_i \text{ is even} \\ 3a_i + 1 & \text{if } a_i \text{ is odd} \end{cases}$$

The Collatz conjecture, which is *unproven*, states that for any starting pint $a \in \mathbb{Z}^+$, the sequence eventually reaches 1.

Write code that accepts an input integer from the user as a_0, and then updates it using the above rule until the sequence reaches 1. Output the entire sequence to the screen. For example, if the user enters 17, the output should look like this:



```
17
52 26 13 40 20 10 5 16 8 4 2 1
Press any key to continue . . .
```

Hints:

- We don't need separate variables for $a_0, a_1, a_2, \ldots$ we can just use one variable which is updated at each step.
- For large enough input values, the intermediate values of the sequence may overflow the range that an `int` can hold, so you could use a `long long` if you want to check such values.

6. Write a program that prompts the user to enter two integers, and outputs their greatest common divisor. Try a straightforward way, and also Euclid's algorithm. As a check on the correctness, the results should be the same for both methods. The Wikipedia page on Euclid's algorithm contains an example with the first few steps, which could also be

used as a check.

Remember, for `int` variables `m` and `k`, `k` is a divisor of `m` if and only if `m%k==0`.

The functions `min` and `max` could be useful here: these need the library `<algorithm>`

```
#include <algorithm>
    ...
    int r = min(m, n);
    ...
```

7. The Schildt book contains many brief sets of questions as a progress check, as well as more questions at the end of each chapter. Checking that you can answer these will help you to check your understanding. We have covered material up to the end of chapter 3, although there are parts in those chapters that we have not covered, so you would need to read the chapters first.