Exercises Day 1

September 28

Introductory Programs

To begin with we will get the compiler up and running and make sure we can compile a simple program. Assuming you are using the GNU compiler for now, you can compile and run your program using the following commands

```
g++ -o program code.cpp
./program
```

One can pass compile flags to the compiler to tell it how it should compile, e.g. how strict it should be with warnings, whether it should compile with debugging support and so on and so forth. Since we will make use of the C++11 syntax, we have to tell the compiler that we are using the new standard. We can do this by calling

```
g++ -o program -std=c++11 code.cpp
./program
```

- Write a program that somehow makes use of the `auto` keyword and have your program compile and run
- Write a program that reads two integers from standard input and prints the sum of the two

Integers and float

- Try multiplying together two numbers so that the result is larger than the largest `int` and store the result in an `int`, what happens?
- Try the same with a floating point type
- Try dividing by zero and print the result

Advanced:

As you all know all data is stored in binary on a computer. For an unsigned 4bit integer this is of course done in the expected way, so that the number 5 is stored as 0101. For floating point numbers, this is somewhat more advanced where parts of the bit-content is dedicated to the decimals, and part of it is dedicated to the exponential factor. You can read more about this on Wikipedia. In this exercise we will try to make a program that writes out the bit-content of the various built in types of C++. This is most easily done with the bitwise operators in C++.

Infinite Loops

- Write an infinite loop that prints the number 42 to the terminal forever. Do this both with a `for` loop and a `while` loop. *Hint: You can stop you program by pressing Ctrl-C*

- Recreate the factorial function from the lecture using an infinite loop and a loop control statement (`break` or `continue`)

Checking for prime

- Write a program that reads an integer from the terminal and checks whether the number is a prime or not
- Make the program a bit more advance, and rather than simply checking whether it is a prime, also print out the primes it can be factorised to

Multi-dimensional arrays

- Write a small program that stores the 20x20 multiplication table in a 20x20 array
- Write code so that the array is printed to the screen in some organised fashion

A game of Mastermind

Finally we will try to write a game of Mastermind. Mastermind is basically a board game where the "codemaker" sets up a combination of 4 colours, and the player will try to guess those in 12 turns. With every guess the codemaker will tell the player how many of their guesses are both the right colour and position, as well as how many additional colours are right, but in the wrong position.

- Write a program that you can play mastermind with