Exercise 1

Tasks marked with a * are assessed coursework. Hand in your solutions to these via email to `rn@ic.ac.uk`*. (Resit students do not need to submit coursework.) Use the subject line "*`C++ CW: surname_firstname_CW1`*", where* `surname_firstname_CW1.cpp` *is the attached file that contains your solution. The course will be assessed based on 5 pieces of coursework (25%) and an end of term driving test (75%). Your submission must be **your own work** (submissions will be checked for plagiarism), and it should compile (and run) with the GNU C++ compiler* `g++`*. The deadline for submitting the coursework is 10pm on **27/01/2019**.*

In all of the following programs you should make use of the STL container class `vector<>`. Remember to `#include<vector>`. Do not use C-style arrays, like `int *`. Make sure that whenever you pass `vector`s to a function, do so by reference, not by value. Similarly, do not return `vector`s as the result of a function. The following extract demonstrates the most important features of the class `vector<>`.

```
vector<int> v;
cout << "size : " << v.size() << endl;               // size :  0

for (unsigned int i = 0; i < 10; ++i)  v.push_back(i);
cout << "size : " << v.size() << endl;               // size : 10

v[3] *= 3;  v[7] = 0;
for (unsigned int i = 0; i < v.size(); ++i)  cout << v[i] << " ";
cout << endl;                                        // 0 1 2 9 4 5 6 0 8 9
```

For more details visit `www.sgi.com/tech/stl/Vector.html` or `www.cppreference.com/cppvector.html`.

1. *Prime numbers*
   Write a program that does all of the following.
   (a) Given p, test if p is prime.
   (b) Given M, calculate all prime numbers $p \leq M$.
   (c) Given n, calculate either the first n or the n^{th} prime number.

   What is the 1,000,000th prime number? What is the percentage of prime numbers up to that number?
   [*Hint: To make your code efficient you should use the fact that p is prime $\iff$ q is not a factor of p for all known prime numbers $q \leq \sqrt{p}$. Note that 2 is the smallest prime number.*]

2*. *Cards*
   Assume you are given a stack of $2k$ cards, $k \in \mathbb{N}$. The cards are numbered from top to bottom from 1 to $2k$, so the topmost card is 1.

   Now the stack of cards is shuffled in the following way. During an m-shuffling ($m \in \mathbb{N}, m \leq k$) the top m cards are taken off the stack. Then – together with the next m cards – they are used in an alternating fashion to build a new stack. The remainder of the stack stays as it is.

   E.g. the initial $k = 8$ stack (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16) looks like this after one 6-shuffling: (7, 1, 8, 2, 9, 3, 10, 4, 11, 5, 12, 6, 13, 14, 15, 16).

   Now a complete shuffling of the stack of $2k$ cards consists of a 1-shuffling, followed by a 2-shuffling, a 3-shuffling, . . ., a k-shuffling (in this order).

   Write a program that asks for $k \geq 2$ and then does a complete shuffling of the stack of $2k$ cards. At the end the program should answer the following questions.
   (a) What are the three topmost cards after the complete shuffling?
   (b) What are the three cards that were on top of the stack most frequently during the shuffling? (Among cards with the same frequency the lower valued one is of higher interest.)
   (c) After which shuffling was the topmost card from (a) on top for the first time?
   (d) How often was the topmost card from (a) on top throughout the shuffling?

   Your submitted program will be run for numbers k between 2 and 10000.

   [*Hint: For k =300 the statistics are (a) 561, 175, 42; (b) 14, 38, 1 [5, 5, 4 times]; (c) 300; (d) 1.*]