General Instructions

1. Download `Practical01.zip` from the course website.

   Extract the file under the `[CppCourse]\[Practicals]` folder. Make sure that the file structure looks like:

```
[CppCourse]
   -> [boostRoot]
   ...
   -> [Practicals]
      -> [Practical01]
         -> Practical01Exercises.hpp
         -> ...
         -> [Src]
            -> Practical01.cpp
```

2. Open the text file `[CppCourse]\CMakeLists.txt`, uncomment the following line by removing the `#`:

   ```
   #add_subdirectory(Practicals/Practical01)
   ```

   and save the file. This registers the project with `cmake`.

3. Run `cmake` in order ot generate the project.

4. The declaration of the functions to be implemented are in `Practical01Exercises.hpp`. Create a `cpp` file for each function and add them to the project.

5. Implement the functions into the newly added `.cpp` file(s) under the `[Src]` folder. Do not modify any of the other files.

6. Compile and run your code (the `main` is provided with the project). If the minimum requirements are met, an output text file `Practical01_output.txt` will be created.

7. You are expected to hand in `Practical01_output.txt` and all the `*.cpp` files you created and put any of your own code into. These files are to be submitted via Moodle.

Exercise 1

```
1 double Norm2(const std::vector<double> & dVec);
```

This function takes a vector $x = (x_1, \ldots, x_n)$ and returns

$$\sqrt{\sum_{i=1}^{n} x_i^2}.$$

Exercise 2

```
double NormInf(const std::vector<double> & dVec);
```

This function takes a vector $x = (x_1, \ldots, x_n)$ and returns

$$\max_{1 \leq i \leq n} |x_i|.$$

Exercise 3

```
double MonteCarlo1(double dR,
                   double dSigma,
                   double dS0,
                   double dK ,
                   double dT,
                   unsigned long int iN);
```

`MonteCarlo1()` takes the risk-free rate, volatility, initial stock price, strike price, time to maturity and the sample size and returns a Monte Carlo estimate of the corresponding European Call option price.
Note: the header file `Utils/UtilityFunctions.hpp` inside the `utils` namespace contains two functions that one might find useful.

```
    void NormalDist(std::vector<double> &vArg);
```

```
    double NormalDist();
```

The first function takes a vector by reference and fills it up with standard normals. The second function takes no argument but returns one single standard normal variable.

Exercise 4

```
std::vector<double> MonteCarlo2(double dR,
                   double dSigma,
                   double dS0,
                   double dT,
                   unsigned long int iN,
                   Payoff call);
```

`MonteCarlo2()` takes the same arguments as `MonteCarlo1` except for the strike, instead it takes a `Payoff` type, where

```
typedef std::function<double(double)> Payoff;
```

that is, `Payoff` wraps functions that take a `double` and return a `double`.
`MonteCarlo2()` returns a vector with two entries; the first one is the MC estimate, and the second one is the estimated standard deviation of the MC estimate.

Exercise 5

```
1 double callAt1(double dS);
```

`callAt1()` is a particular function of type `Payoff` implementing a European payoff with strike 1.