Johns Hopkins University
Department of Applied Mathematics & Statistics
Lecturer: Maxim Bichuch

Fall 2015

EN 443
Financial Computing in C++
Assignment 3

due on Wed, Sep 23, 1:30pm

1. Implement the strcat function:
   ```
   char * myStrcat ( const char * st1, const char * st2 );
   ```
   to concatenate two character array. Assuming the character array st1, has enough elements to also contain st2, append a copy of the character array st2 to it. The function should return the resulting character array. For this problem, please do not use any build in functions/classes.

2. Implement a function that performs bubble sort algorithm. In this algorithm, during the i^{th} iteration, $1 \leq i \leq n$, assuming that the largest $i-1$ elements have already been found and are fixed in their places $1, .., i-1$, the algorithm goes over the other $n-i+1$ elements, finds the largest one, and swaps it with the element in the i^{th} place. The function interface should be
   ```
   void myBubbleSort (double* arr, int size);
   ```

3. Implement the quick-sort algorithm to sort an array of doubles. The algorithm is as follows

   (a) Pick an element, called a pivot, from the array.

   (b) Reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.

   (c) Recursively sort the sub-array of lesser elements and the sub-array of greater elements.

   Check your function with an array of some constant size N populated by random numbers. Use the following functions to generate random numbers

   ```
   void srand ( unsigned int ); // initialize random number generator,
   int rand(void); // returns a random number between 0 and RAND_MAX.
   ```

   For example, the code below prints a uniformly distributed number between 0 and 1.

```
#include <iostream >
#include <cstdlib >
#include <ctime >
using namespace std;
int main()
{
        srand ( time(NULL) );                                   //initialize random seed
        cout<< rand() /(static_cast<double> (RAND_MAX)) ;       // generate random number
        cout<<endl;
        return 0;
}
```

4. Implement two functions that compute the Black-Scholes price of a call and put options. Recall the Black-Scholes the prices of a call and put options with time to maturity τ are given respectively as:

$$C = S\Phi(d_+) - K\,\mathrm{e}^{-r\tau}\,\Phi(d_-),$$
$$P = K\,\mathrm{e}^{-r\tau}\,\Phi(-d_-) - S_0\Phi(-d_+).$$

where Φ is the standard normal c.d.f.,

$$d_\pm = \frac{\log\left(\frac{S}{K}\right) + \left(r \pm \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}},$$

and where S is the current stock price, K is the strike, r is the interest rate, σ is the volatility.

```
double BlackScholesCall (double S, double K, double sigma, double tau, double r );
double BlackScholesPut (double S, double K, double sigma, double tau, double r );
```

Compute the normal c.d.f Φ by using the following approximation for $x \geq 0$

$$\Phi(x) = 1 - \phi(x)\sum_{n=1}^{5} b_n t^n,\ t = \frac{1}{1 + b_0 x},$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}}\,\mathrm{e}^{-\frac{x^2}{2}}$ the normal p.d.f., and

$$b_0 = 0.2316419,$$
$$b_1 = 0.3193815300,$$
$$b_2 = -0.356563782,$$
$$b_3 = 1.7814779370,$$
$$b_4 = -1.821255978,$$
$$b_5 = 1.3302744290.$$