Project 1: Performance Measurement

There are at least two different algorithms that can compute X^N for some positive integer N.

Algorithm 1 is to use $N - 1$ multiplications.

Algorithm 2 works in the following way: if N is even, $X^N = X^{N/2} \times X^{N/2}$; and if N is odd, $X^N = X^{(N-1)/2} \times X^{(N-1)/2} \times X$. Figure 2.11 in your textbook gives the recursive version of this algorithm.

Your tasks are:

(1) Implement Algorithm 1 and an *iterative* version of Algorithm 2;

(2) Analyze the complexities of the two algorithms;

(3) Measure and compare the performances of Algorithm 1 and the iterative and recursive implementations of Algorithm 2 for X=1.0001 and N = 1000, 5000, 10000, 20000, 40000, 60000, 80000, 100000.

To measure the performance of a function, we may use C's standard library **time.h** as the following:

```
#include <time.h>
clock_t   start, stop; /* clock_t is a built-in type for processor time (ticks) */
double    duration;    /* records the run time (seconds) of a function */

int main ( )
{    … …
     /* clock() returns the amount of processor time (ticks) that has elapsed
        since the program began running */
     start = clock();    /* records the ticks at the beginning of the function call */
     function();         /* run your function here */
     stop = clock();     /* records the ticks at the end of the function call */
     duration = ((double)(stop - start))/CLK_TCK;
     /* CLK_TCK is a built-in constant = ticks per second */
     … …
     return 1;
}
```

Note: If a function runs so quickly that it takes less than a tick to finish, we may repeat the function calls for K times to obtain a total run time, and then divide the total time by K to obtain a more accurate duration for a single run of the function. The

repetition factor must be large enough so that the number of elapsed ticks is at least 10 if we want an accuracy of at least 10%.

The test results must be listed in the following table:

	N	1000	5000	10000	20000	40000	60000	80000	100000
Algorithm 1	Iterations (*K*)								
	Ticks								
	Total Time (sec)								
	Duration (sec)								
Algorithm 2 (iterative version)	Iterations (*K*)								
	Ticks								
	Total Time (sec)								
	Duration (sec)								
Algorithm 2 (recursive version)	Iterations (*K*)								
	Ticks								
	Total Time (sec)								
	Duration (sec)								

The performances of the three functions must be **plotted** in the **same** *N*–run_time coordinate system for illustration.

Grading Policy:

This assignment is due Monday, October 10th, 2016 at 10:00pm.

- **Programmer:** Implement the three functions **(30 pts.)** and a testing program **(20 pts.) with sufficient comments**.
- **Tester:** Decide the iteration number *K* for each test case and fill in the table of results **(8 pts.)**. Plot the run times of the functions **(12 pts.)**. Write analysis and comments **(10 pts.)**.
- **Report Writer:** Write Chapter 1 **(6 pts.)**, Chapter 2 **(12 pts.)**, and finally a complete report **(2 pts. for overall style of documentation)**.