


**CS244 Data Structure
Programming Assignment 4
(100 points)
No late submission will be allowed!**

Objective:

- Practice on sorting.

Problem Statement

Writing a program that first randomly generates 5,000,000 numbers (integers) that are in the range [0, 5,000,000], using an array or vector to hold the number, and then using quicksort as following steps:

1. Sort the array using the middle element of the array and subarrays as the pivots.
2. Sort the array using the median of the first, last and middle elements of the array and subarrays as the pivots.
3. Sort the array using the middle element of the array and subarrays as the pivots. However, when the size of any subarrays reduces to less than 20, sort the subarray using insertion sort (insertion sort has to be implemented separately and then called inside the quick sort). 
4. Sort the array using the median of the first, last and middle elements of the array and subarrays as the pivots. However, when the size of any subarray reduces to less than 20, sort the subarray using insertion sort (insertion sort has to be implemented separately and then called inside the quick sort).
5. Calculate and print the CPU time for each of the preceding four steps.

Your program must compile and execute without any errors or warnings, failing to implement your program according the instructions may result in zero.

Sample output:

```
Quick sort time, with pivot middle element: 3 seconds
Quick sort time, with pivot median element: 3 seconds
Quick sort and insertion sort time, with pivot middle element: 3 seconds
Quick sort and insertion sort time, with pivot median element: 3 seconds
```

Supplemental Information:

1. To record running time, you need to record starting time and ending time, then calculate the running time by subtracting the starting time with the ending time. Example code for calculating the running time for the function "quicksort()":

```
#include <ctime>
clock_t startTime, endTime;
startTime = clock();
quicksort();
endTime = clock();
cout<<"Quick sort time: "<<(endTime -startTime)/
    CLOCKS_PER_SEC <<" seconds" << endl;
```

2. To generate 5,000,000 numbers within range [0, 5,000,000], the following example code can be used as reference.

```
#include <ctime>
srand(time(0));
int i = 0;
int list[5000000];
while (i < 5000000)
{
    List[i] = (rand() + time(0)) % 5000000;
    i++;
}
```

Submission Guidelines:

To receive full credit, you must follow these guidelines

1. All code must include comments, be properly indented and use descriptive variable names where appropriate.
2. Compress the entire project folder with solution directory to a compressed file and submit this to the appropriate dropbox on D2L
3. After submission, double check to make SURE you submit the proper files, **YOU WILL NOT HAVE A CHANGE TO FIX THIS AFTER THE DUE DATE!!**
Please ask for help if you need it.

Grading:

1. If your program does not compile and lack efforts (i.e., lack comments and function implements), you receive 0.
2. If your program does not compile but shows significant efforts in code, you receive at most 30% of the total credit.
3. If you program can compile but missing functionalities (i.e., nice interface), you receive about 60% of the total credit. You may receive more than 60% according to the following rubrics.

	Points	Deducted Points
Correctness	75	
1. Correctly implement step 1	15	
2. Correctly implement step 2	15	
3. Correctly implement step 3	15	
4. Correctly implement step 4	15	
5. Correctly implement step 5	15	
Style	25	
Lay out your program in a readable fashion and user-friendliness	10	
Include comments for each function	15	

Provide separate .cpp files and header files for class(es)	5	
Your Score		