School of Computer Science, University of Windsor

COMP 2540: Data Structures and Algorithms
Term: Summer 2021
Instructor: Dr. Asish Mukhopadhyay

Lab 7
Posted: 02 July., 2021
Due: 11:59 pm, 08 July, 2021

Instructions:

- You are expected to finish the lab by the end of the posted date. Submissions beyond the due date will earn a penalty of $n * 25\%$, where $n = submissionDay - dueDay$. Thus if the lab is due Tuesday and you submit on Wednesday, this will be considered a day late.

- Whether or not you finish your work during the lab period, you will have to upload a script file of your work on BLACKBOARD for record-keeping and grading before the beginning of the next lab. Create a script file as follows:

```
1. script LabName.txt
2. cat LabName.c
3. cat input.txt
4. cc labName.c
5. ./a.out < input.txt
6. ls -l
7. exit (DO NOT FORGET THIS STEP!!)
```

- There will be no make-up for missed labs. If you have missed a lab for truly extenuating circumstances (like illness or family emergency) I will consider allowing you to make a late submission. However, I need to be informed by email about this on the day of the missed lab. The email should include your name and SID.

Problem:

Implement both the insertionSort and quickSort algorithms, as described in the pseudocodes in the courseware. Write functions for the two sorting routines and call them from the main program. Also, write the partition method that lies at the heart of quickSort as a separate function that is called from the quickSort routine.

You program should take a list-size from the user, populate an array of this size with random integers and output the sorted list to the screen for small values of list-size (so that we can check if your program has bugs). To demonstrate the difference in the running times of insertionSort and quickSort, the list-size must be large (you will see that quickSort runs much faster). In this case, you should output to the screen only the running times.

To aid you further in completing this lab, here is a framework that you might want to follow to write a complete program :

```
//includes

//function prototypes

void quickSort(int list[], int, int);
void insertionSort(int list[], int);
int partition(int list[], int, int);

int main(void){

//call quickSort
//call insertionSort


}//end main

//insertionSort
void insertionSort(int a[], int listSize ) {


}//end insertionSort

//quickSort
```

```
void quickSort(int a[], int p, int q) {


}//end quicksort


// the partition procedure is somewhat tricky to implement;
// you are free to implement your own
// partition the segment of the array a[], going from index p to index q
int partition(int a[], int p, int q){

        int i = p-1;
        int j = q+1;
        int x = a[p];

        int flag = 1;

        while(flag){

                // move left pointer
                do i = i+1;
                while (a[i]<x);

                //move right pointer
                do j = j-1;
                while (a[j]>x);

                if (i<j){
                //exchange a[1] and a[j]
                }
                else  flag = 0;
        }//end while
return j;

} // end partition
```

(10 points)