National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)
CS2092D Programming Laboratory
Assignment #3

Submission deadline (on or before):

- 23.09.2020, 10:00 PM

Policies for Submission and Evaluation:

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors in the Linux platform.
- During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
- Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz). The name of this file must be

$$ASSG < NUMBER > \_ < ROLLNO > \_ < FIRST - NAME > .zip$$

(For example: $ASSG1\_BxxyyyyCS\_LAXMAN.zip$). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

- The source codes must be named as

$$ASSG < No. > \_ < ROLLNO > \_ < FIRST - NAME > \_ < PROGRAM - No. > .c$$

(For example: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of 0 marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

- Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

General Instructions

- Programs should be written in C language and compiled using C compiler in Linux platform. **Submit the solutions to questions 1 and 2 through the submission link in Eduserver**.

The Sorting Problem can be formally stated in terms of the input/output relationship as follows:

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$

Output: A permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

QUESTIONS

General Note: For the following programs, do not declare the array as a global variable. You may pass the arrays as function arguments by using the concept of pointers.

1. Write a program that uses the INSERTION-SORT algorithm for sorting a given sequence of integers present in an array A and prints the number of comparisons performed during sorting. Your program must contain the following functions.

   - INSERTION-SORT(A) - A function that takes as input an array A and sorts it using insertion sort.
   - PRINT(A) - A function that takes as input an array A and prints its contents in order, with a single space separating the elements. This function should only be called from the MAIN() function.

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^4]$, the size of the array A.
   - The second line lists the n elements in A, as space-separated integers in the range $[-1000, 1000]$.

   Output Format:

   - The first line of the output contains the elements of A in non-decreasing sorted order, with a single space separating each element.
   - The second line of the output contains the number of comparisons performed (between the elements of A) during sorting.

   Note:

   The number of comparisons made can vary a little depending on the way Insertion Sort is implemented. As such, we will be considering the number of comparisons as per the Insertion Sort algorithm given in CLRS.

   Sample Input 1:
   10
   23 76 89 3 8 0 789 123 2789 25

   Sample Output 1:
   0 3 8 23 25 76 89 123 789 2789
   17

   Sample Input 2:
   10
   90 89 78 67 56 45 34 23 12 11

   Sample Output 2:
   11 12 23 34 45 56 67 78 89 90
   45

2. Write a program that uses the MERGE-SORT algorithm for sorting a given input sequence of integers present in an array A and prints the number of comparisons performed during sorting. Your program must contain the following functions. (In what follows, the notation $A[p..r]$ denotes the sub-array of A, contained within the p^{th} and r^{th} indices, both inclusive.)

   - A recursive function MERGE-SORT($A,\ p,\ r$) that takes as input an array A and sorts the elements in the sub-array $A[p..r]$.
   - A function MERGE($A,\ p,\ q,\ r$) that takes as input an array A in which the sub-arrays $A[p..q]$ and $A[q+1..r]$ are sorted. It then merges these sub-arrays such that the the sub-array $A[p..r]$ is sorted.
   - PRINT(A) - A function that takes as input an array A and prints its contents in order, with a single space separating the elements. This function should only be called from the MAIN() function.

   Input format:

   - The first line of the input contains an integer $n \in [0, 10^5]$, the size of the array A.
   - The second line lists the n elements in A, as space-separated integers in the range $[-10^3, 10^3]$.

   Output Format:

   - The first line of the output contains the elements of A in sorted order, separated by space.
   - The second line of the output contains the number of comparisons performed during sorting.

   Notes:

   - The number of comparisons made can vary a little depending on the way Merge Sort is implemented. As such, we will be considering the number of comparisons as per the Merge Sort algorithm given in CLRS.
   - In particular, to split an array $A[p..r]$ into two sub-arrays, the MERGE-SORT() function should compute an index $q \in [p, r]$ such that $A[p..q]$ contains $\lceil n/2 \rceil$ elements, and $A[q+1..r]$ contains $\lfloor n/2 \rfloor$ elements).

   Sample Input 1:
   10
   23 76 89 3 8 0 789 123 2789 25

   Samle Output 1:
   0 3 8 23 25 76 89 123 789 2789
   34

   Sample Input 2:
   10
   90 89 78 67 56 45 34 23 12 11

   Sample Output 2:
   11 12 23 34 45 56 67 78 89 90
   34