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):

- 08.09.2021, 9:00 AM

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 using gcc compiler.
- 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`

  (Example: *ASSG*1_*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<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG*1_*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 gcc compiler. **Submit the solutions to the questions through the submission link in Eduserver**.
- Check your programs with sufficiently large values of inputs with in the range as specified in the question.
- Global and/or static variables should not be used in your program.

QUESTIONS

1. Write a program that reads n distinct integers of an array A, given in ascending order and checks whether a given integer k is present in the array using recursive binary search. If k is present in the array, print its index in A. Otherwise, print -1. Also print the number of calls made to the recursive binary search function over the course of the program (You are permitted to use a global variable to maintain this count).

   Assume that the array index starts from 0.

   Input Format:

   - The first line contains an integer, $n \in [0, 10^5]$, the size of the array A.
   - The second line lists the n distinct elements in A, as space-separated integers in the range $[-10^6, 10^6]$.
   - Third line is an integer $k \in [-10^6, 10^6]$ to be searched in the array.

   Output Format:

   - The first line contains the index of k, if it is present in the array A. Otherwise print -1.
   - The second line contains the number of calls to the recursive function made over the course of the program.

   Sample Input 1:
   7
   12 35 50 59 60 73 90
   73

   Sample Output 1:
   5
   2

   Sample Input 2:
   7
   12 35 50 59 60 73 90
   100

   Sample Output 2:
   -1
   3

2. Modify your program for Question 1 such that it reads n integers (not necessarily distinct) of an array A, given in ascending order and checks whether a given integer k is present in the array using recursive binary search. If k is present in the array, print the index of its first occurrence in A. Otherwise, print -1. Also print the number of calls to recursive function made over the course of the program.

   Note: You should find the first occurrence of k using recursive calls only, and should not linearly traverse the array for any purpose.

   Assume that the array index starts from 0.

   Input Format:

   - The first line contains an integer, $n \in [0, 10^6]$, the size of the array A.
   - The second line lists the n elements in A, as space-separated integers in the range $[-1000, 1000]$.
   - Third line is an integer $k \in [-1000, 1000]$ to be searched in the array.

Output Format:

- The first line contains the index of the first occurrence of k, if k is present in the array A. Otherwise print -1.
- The second line contains the number of calls made to the recursive function over the course of the program.

Sample Input 1:
7
12 35 50 59 60 73 90
73

Sample Output 1:
5
2

Sample Input 2:
15
1 1 2 2 2 2 2 2 2 3 3 3 3 3 3
3

Sample Output 2:
9
3

3. Given an unsorted array A containing m distinct integers and a sorted array B containing n integers, write a program to find their intersection using recursive binary search. Intersection of two arrays is the list containing common elements from both the arrays. If an element is repeated, then only one occurrence of that element should be present in the intersection. Assume that the array index starts from 0.

Input Format:

- The first line contains two space-separated integers m and $n \in [0, 10^6]$, the size of the arrays A and B, respectively.
- The second line lists the m distinct elements in A, as space-separated integers in the range $[-10^3, 10^3]$.
- The third line lists the n elements in B, sorted in descending order, as space-separated integers in the range $[-10^3, 10^3]$.

Output Format:

- The output contains the intersection of arrays A and B as space-separated integers in their order of occurrence in A.

Sample Input 1:
7 5
12 35 23 59 15 73 90
95 73 67 59 3

Sample Output 1:
59 73

Sample Input 2:
4 6

33 25 56 19 77 67
6 5 4 3 2 1

Sample Output 2:
-1

4. 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.
   - A function PRINT(A, n) that takes as input an array A of size n, 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 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 contains the elements of A in sorted order, separated by space.
   - The second line 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 0 789 123 889 25

   Samle Output 1:
   0 3 8 23 25 76 89 123 789 889
   21

   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
   15

5. Write a program to count the number of inversions in an array A containing n distinct integers by modifying the MERGE-SORT algorithm. The inversion count of an array indicates how far (or close) the array is from being sorted; if the array is already sorted, then the inversion count is 0, whereas the count will be maximum if the array is sorted in the reverse order. Two elements $A[i]$ and $A[j]$ form an inversion if $A[i] > A[j]$ and $i < j$.

Input Format:

- The first line contains an integer $n \in [0, 10^5]$, the size of the array A.
- The second line lists the n distinct elements in A, as space-separated integers in the range $[-10^6, 10^6]$.

Output Format:

- The first line should contain the number of inversions in the array A.
- The second line should contain the number of comparisons performed over the course of the program (Refer the note given in Question 4).

Sample Input 1:

10
23 76 89 3 8 0 789 123 889 25

Samle Output 1:

17
21

Sample Input 2:

10
90 89 78 67 56 45 34 23 12 11

Sample Output 2:

45
15