National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)
CS2092D Programming Laboratory
Assignment #2

Submission deadline (on or before):

- 15.08.2022, 11:55 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 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: $ASSG1\_BxxyyyyCS\_LAXMAN.zip$). 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<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.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 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.
- Consider array indexes start at 0, unless otherwise specified.

QUESTIONS

1. Write a program to find the product of frequencies of all the distinct elements present in the given string. Consider maximum possible length of the string as 1000.
   For example, in the word `mississippi`, the letter `m` occurs once, `i` appears four times, `s` occurs four times, and `p` occurs twice. Therefore the expected output is $1 * 4 * 4 * 2 = 32$.

   Input format:

   - The first line of the input is a string with lowercase characters $[a - z]$.

   Output format:

   - The output is the product of the frequencies of the distinct elements present in the given character array.

   Sample Test Case #1

   Input: honeybeen

   Output: 6

   Sample Test Case #2

   Input: aaabbbccc

   Output: 27

2. Given an array of size n, which could be split into m non-overlapping sub-arrays each of size k. (It is guaranteed that size of array n is always $m * k$). Now, write a program to calculate the number of such non-overlapping sub-arrays having element x in it.
   For example, we are asked to split an array A = 1 2 3 1 2 5 1 7 8 5 1 9 with 12 elements into non overlapping sub-arrays each of size $k = 3$, then $m = 4$. So, the possible sub-arrays are: 1 2 3, 1 2 5, 1 7 8, 5 1 9. If given x as 1, the expected output is 4, which is the number of sub-arrays having element 1 in it.

   Input format:

   - The first line of the input contains two integers $n \in [1, 1000]$ and $k \in [1, n]$ separated by space.
   - The second line lists the n elements in the array A, as space-separated integers in the range $[-1000, 1000]$.
   - The third line contains an integer $x \in [-1000, 1000]$.

   Output format:

   - Print an integer, which is the number of non-overlapping sub-arrays of size k having element x. If no sub-array contains element x print 0.

   Sample Test Case #1

   Input:
   6 2
   98 90 -14 31 -46 56
   90

   Output: 1

   Sample Test Case #2

   Input:
   9 3
   22 -12 14 23 -34 15 -12 14 17
   12

   Output: 0

3. Write a program to delete all occurrences of the second largest element in a given array A. (You should not declare and use any array other than A). If no second largest element exists in the array, your program will return A.

Input format:

- The first line of the input contains an integer $n \in [1, 10^3]$, 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 output is the array after deleting all occurrences of the second largest element.

Sample Test Case #1

Input:
8
1 5 6 9 8 3 8 5

Output:
1 5 6 9 3 5

Sample Test Case #2

Input:
5
10 10 10 10 10

Output:
10 10 10 10 10

4. Write a program to find the indices of all the elements greater than the given target element t in the array A sorted in non-decreasing order. Use the idea of Binary Search to find the index of the target and then perform Linear Search to find the other required indices.
Note :Follow the binary search algorithm taught in class.

Input format:

- The first line of the input contains two integers $n \in [1, 10^3]$, the size of the array A and $t \in [-1000, 1000]$.
- The second line lists the n sorted elements in A, as space-separated integers in the range $[-1000, 1000]$.

Output format:

- The first line of the output is the index of the target found by the Binary Search Algorithm or -1 if the target is not present in the array.
- The second line of the output contains the indices of all the elements greater than the given target element or -1 if the target or any element greater than the target is not present in the array.

Sample Test Case #1

Input:
8 14
12 13 14 14 14 15 15 16

Output:
3
5 6 7

Sample Test Case #2

Input:
7 -17
-22 -22 -20 -20 -18 -18 -17

Output:
6
-1

Sample Test Case #3

Input:
5 106
100 102 104 108 110

Output:
-1
-1