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):

- 22.08.2022, 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

- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$).

  Create separate files for each question and place all of them in a single folder named

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>`

  (Example: $ASSG1\_BxxyyyyCS\_LAXMAN$). Then Zip the **folder**.

- Submit the single ZIP (.zip) file you have created(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.

  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.
- You shouldn't change the function prototype specified in the question.
- **Consider array indexes start at 0, unless otherwise specified.**

QUESTIONS

1. You are given an array A consisting of n real numbers. All the elements in this array are guaranteed to be unique. For each element k in array A, you need to find the position index of k where it would have appeared if A was a sorted list using **Insertion Sort**.

   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 floating point numbers in the range $[-1000, 1000]$.

   Output format:
   - Print n space-separated integers on a single line which indicates the positions for each element in A, where it would have existed if A was a sorted list.

   Sample Test Case #1
   Input:
   ```
   5
   1.25 2.56 3.08 4.54 5.9
   ```

   Output:
   ```
   0 1 2 3 4
   ```

   Sample Test Case #2
   Input:
   ```
   7
   5.26 2.06 4.65 6.78 1.3 10.07 8.654
   ```

   Output:
   ```
   3 1 2 4 0 6 5
   ```

2. You are given two integer arrays, A and B, with distinct elements, sorted in non-decreasing order, with sizes m and n, respectively, and another integer, *target*. Write a program to find the index of the *target* after **merging** arrays A and B in the non-decreasing sorted order. Return -1 if the *target* is not present.

   Input format:
   - The first line of the input contains two space-separated integers $m \in [1, 10^4]$ and $n \in [1, 10^4]$ representing the size of arrays A and B.
   - The second line lists m elements in A, as space-separated integers in the range $[-1000, 1000]$.
   - The third line lists n elements in B, as space-separated integers in the range $[-1000, 1000]$.
   - The fourth line contains the *target* value.

Output format:

- Print an integer that represents the index of *target* if present, otherwise -1.

Sample Test Case #1
Input:

```
4 4
1 3 5 7
2 4 6 8
4
```

Output:

```
3
```

Sample Test Case #2
Input:

```
3 2
1 3 4
2 6
7
```

Output:

```
-1
```

3. Some children are standing in line to buy candies. The candy man will distribute the sweets only if all the children in the line are standing in alphabetical order of their names. Write a program to arrange all the children in the line in the required order so that they get candies. Use the **Merge Sort** algorithm to solve the problem.

   Input format:

   - The first line of the input contains an integer $n \in [1, 10^2]$, indicating the total number of children standing in the line.
   - The following n lines contain the children's names as strings with characters $[A - Z, a - z]$ and space.

   Output format:

   - Print the sorted list of student names in n lines.

   Sample Input:

   ```
   5
   Manoj Kumar
   Ranjan Ram
   Alok Ravi
   Akash Singh
   Alan Thomas
   ```

   Sample Output:

   ```
   Akash Singh
   Alan Thomas
   Alok Ravi
   Manoj Kumar
   Ranjan Ram
   ```

4. Given an array A of n distinct integers. You have to perform two tasks. Firstly, find the sum of digits of each element in the given array. Secondly, perform **insertion sort** on the array. The sorting should be done in such a way that the sorted list has to be in accordance with the sum of digits of each array element. For example, consider we have a 3-element array A = 123, 342,

400. Then the expected sorted output would be 400, 123, and 342. (Since (4+0+0) < (1+2+3) < (3+4+2)). If multiple elements yield the same sum, whichever element comes first in the input should also come first in the output.

Now, write a menu-driven program that implements the above scenario. Your program should contain the following functions:

- *main*(): Repeatedly read a character '*r*', '*p*', '*d*', '*s*', or '*t*' from console and call the corresponding functions, as described below, until character '*t*' is encountered.
- *read_and_store*(A, n): Read n integers and store it in array A.
- *print*(A, n): Print the elements in the array A of size n, each element separated by a space.
- *digit_sum*$(A[i])$: Find the sum of the digits of the element $A[i]$ and return it.
- *digit_sort*(A, n): The function creates the sorted list in accordance with the sum of digits of each array element as described above. This function should use the *digit_sum*() function defined above.

Input/Output format:
The input consists of multiple lines, listing at most 1000 operations to be performed on A. Each line may contain a character from {'*r*', '*d*', '*s*', '*p*', '*t*'} followed by zero or more integers. The integers, if given, are in the range $[1, 10^5]$.

- Character '*r*' : Character '*r*' will be followed by a positive integer n representing the number of elements to be stored in the array A. The next line contains n space-separated integers. Read and store these elements in A using the *read_and_store*() function.
- Character '*p*' : Print the elements in the array A using the *print*() function.
- Character '*d*' : Character '*d*' will be followed by a **valid** integer input $i \in [0, n-1]$ which is the index of an array element whose sum of digits needs to be calculated.
- Character '*s*' : On receiving character '*s*' , perform the required sorting operation using the *digit_sort*() function.
- Character '*t*' : Terminate the program with return -1.

Sample Input:
```
r 7
215 401 521 165 911 404 122
p
d 3
s
p
d 3
t
```

Sample Output:
```
215 401 521 165 911 404 122
12
401 122 215 521 404 911 165
8
-1
```