INDIAN STATISTICAL INSTITUTE

MTech(CS) I year 2025-2026

Subject: Computing Laboratory

Lab Test 1 (February 19, 2026)

Total: 60 marks Duration: 4 hours

GENERAL INSTRUCTIONS

1. All programs should take the required input via the standard input (terminal/keyboard), and print the desired output to the terminal.
2. Please make sure that your programs adhere strictly to the specified input and output format. Do not print extra strings asking the user for input, debugging messages, etc. These will cause the automatic checking system to fail.

Q1. (5 marks)

You are given a list of n integers, a_0, a_1, ..., a_{n-1}. First, write a program to just print the integers a_0, a_1, ..., a_{n-1} in ascending order. You may use any sorting algorithm for this problem.

Input format: The input will be of the form n a_0 a_1 ... a_{n-1} (the numbers will be separated by spaces and/or newlines).

Output format: Your program should print a_{i_0} a_{i_1} ... $a_{i_{n-1}}$ where a_{i_0} a_{i_1} ... $a_{i_{n-1}}$ corresponds to a_0 a_1 ... a_{n-1}, but with the integers printed in ascending order.

Q2. (15 marks)

Now write a program, which, given a **sorted** list of integers a_0, a_1, ..., a_{n-1}, and a target T (also an integer), determines whether there exists a_i, a_j with $0 \leq i < j < n$ such that $a_i + a_j = T$. You will get full credit if the running time of your solution is $O(n)$, almost full credit if it is $O(n \log n)$, partial credit if it is $O(n^2)$, but no credit if it is any worse.

Input format: The input will be of the form n a_0 a_1 ... a_{n-1} T (the numbers will be separated by spaces and/or newlines).

Output format: Your program should print a_i and a_j on one line, or the message `Does not exist`. If there are multiple pairs that add up to T, you should print the pair for which a_i is the smallest.

Sample input 1: `5 1 1 1 1 1 2`

Sample output 1: `1 1`

Sample input 2: `8 -314 -100 -8 12 23 41 87 230 130`

Note that the input numbers (-314, -100, ..., 230) are sorted.

Sample output 2: `-100 230`

Q3. (20 marks)

You are given an array $A[0 \ldots n-1]$ that stores the height of n buildings in Baranagar, indexed from East to West. Building i is said to have a *good view* of the Ganges iff every building to the West of i is shorter than building i. Write a program that takes A as input and outputs the indices of buildings that have a *good view*. You will get full credit iff the running time of your solution is $O(n)$.

Input format: The input will be of the form n $A[0]$ $A[1]$ $\ldots$ $A[n-1]$ (the numbers will be separated by spaces and/or newlines).

Output format: The indices (if any) of buildings with good views.

Sample input 1: `5 5 4 3 2 1`

Sample output 1: `0 1 2 3 4`

Sample input 2: `5 1 2 3 4 5`

Sample output 2: `4`

Q4. (20 marks) This problem tries to generalise Exercise 2 of Stacks, Queues, Lists. You have to implement a data structure SEQUENCE, that holds an ordered list of **possibly heterogeneous** elements. Each element may be either an `int`, a `float`, or a null-terminated string (`char *`); all the elements in a SEQUENCE **do not have to be of the same type**.

You may use the following structure to store each element of a sequence.

```
typedef struct {
    int type;
    void *data;
} ELEMENT;
```

Define a suitable integer encoding for the 3 permitted types, e.g., use 1 to represent `int`s, 2 to represent `float`s, and 3 to represent strings. You may modify the above definitions as needed. Your implementation should support the following operations.

- `void print_element(SEQUENCE *s, size_t i)`: prints the i-th element of the sequence pointed to by `s`, if it exists. If the i-th element does not exist, the function should print `Invalid index` as the error message.
- `void get_value(SEQUENCE *s, size_t i)`: if the i-th element of the sequence pointed to by `s` exists, its 'value' (an integer, see the definition below) should be printed; otherwise, the function should print `Invalid index` as the error message.

  Definition: The *value* of an integer n is n itself; the value of a floating point number f is the integer nearest to f; the value of a string s is the sum of the ASCII values of all the characters contained in s (the numerical value of an empty string is defined to be 0).
- `int append(SEQUENCE *s, int type, void *data)`: appends the given data to the given sequence. If the append operation is successful, the length of the sequence should increase by 1, the newly appended data should be accessible as the last element of the sequence, and the function should return 0; otherwise, the function should return -1.

- `void length(SEQUENCE *s)`: prints the number of elements in the sequence pointed to by `s`.
- `void minimum(SEQUENCE *s)`: calls the `print_element()` function on the element that has the smallest value (defined above) in the sequence pointed to by `s`.

Input format: The input will be a list of the above operations, one per line. The number of operations will **NOT** be specified in advance. For `print_element` and `get_value`, the operation name will be followed by a space, and then the index `i`. For `append`, the operation name will be followed by a single space, then a single character (`i` for `int`s, `f` for `float`s, and `s` for strings), followed by another space, and a datum of the appropriate type. You may assume that the strings will consist of no more than 60 non-whitespace characters.

Output format: You should start with an empty sequence (say `S`), perform the given operations one by one on `S`, and print any resulting output. The output of each command should appear on a separate line. Floating point numbers should be stored as `float`s, and printed correct to 6 places of decimal (the default).

Sample input:

```
print_element 10
append s IndianStatisticalInstitute
append i 0
length
append f 3.14159
get_value 1
append i -100
print_element 0
minimum
length
```

Sample output :

```
Invalid index
2
0
IndianStatisticalInstitute
-100
4
```