**ASSIGMENT NO.-10(Searching/Sorting/Linked List/Queue)**

1. WAP to implement Linear Search.

You are given an array A of size N. You are given number K which you have to search in the array and return the index of that element. If there are multiple matches then tell the index of the first matched element. If there is no match, print -1.

**Input Format**

First line of input contains two value ,N and K, where N is the size of the array and K is the key to be searched. Next N lines consist elements of array as shown in sample input

**Constraints**

1 ≤ N ≤ 100000

0 ≤ A[i] ≤ 10^9

'long int' is used for storing number in the range of 10^9

**Output Format**

Print the index of first appearance of element (if found) else -1.

**Sample Input 0**

5 66

11

33

55

66

77

**Sample Output 0**

3

1. WAP to search an element using Binary Search Method.

You need to find index (0-based) of a given key in a sorted array. Use only Binary Search.

**Input Format**

The first line contains N.Next line contains N integers of the array. The next line contains the key to be searched. If element is not found print -1.

**Constraints**

N<=1000000

**Output Format**

Single Interger index or -1.

**Sample Input 0**

5

1 2 3 4 5

4

**Sample Output 0**

3

1. WAP to sort an array of given numbers using selection sort method.

Consider an Array a of size N Iterate from 1 to N In ith iteration select the i th minimum and swap it with a[i]

You are given an array a, size of the array N and an integer x. Follow the above algorithm and print the state of the array after x iterations have been performed.

**Input Format**

The first line contains two integer N and x denoting the size of the array and the steps of the above algorithm to be performed respectively. The next line contains N space separated integers denoting the elements of the array.

**Constraints**

1<=N<=100 1<=a[i]<=100 1<=x<=N

**Output Format**

Print N space separated integers denoting the state of the array after x steps

**Sample Input 0**

5 2

1 2 3 4 5

**Sample Output 0**

1 2 3 4 5

1. WAP to sort an array of given numbers using bubble sort method.

You have been given an array A of size N . you need to sort this array non-decreasing oder using bubble sort. However, you do not need to print the sorted array . You just need to print the number of swaps required to sort this array using bubble sort

**Input Format**

The first line consists of a single integer N denoting size of the array. The next line contains N space separated integers denoting the elements of the array.

**Constraints**

1<=N<=100 1<=a[i]<=100

**Output Format**

Print the required answer in a single line

**Sample Input 0**

5

1 2 3 4 5

**Sample Output 0**

0

1. WAP to sort an array of given numbers using Insertion sort

You have been given an A array consisting of N integers. All the elements in this array are guaranteed to be unique. For each position i in the array A you need to find the position A[i] should be present in, if the array was a sorted array. You need to find this for each i and print the resulting solution.

**Input Format**

The first line contains a single integer N denoting the size of array A. The next line contains N space separated integers denoting the elements of array A.

**Constraints**

1<=N<=100 1<=A[i]<=100

**Output Format**

Print N space separated integers on a single line , where the Ith integer denotes the position of if this array were sorted.

**Sample Input 0**

5

1 2 3 4 5

**Sample Output 0**

1 2 3 4 5

1. WAP to sort an array of given numbers Quick sort method.

Quick sort is based on the divide-and-conquer approach based on the idea of choosing one element as a pivot element and partitioning the array around it such that: Left side of pivot contains all the elements that are less than the pivot element Right side contains all elements greater than the pivot

It reduces the space complexity and removes the use of the auxiliary array that is used in merge sort. Selecting a random pivot in an array results in an improved time complexity in most of the cases.

**Input Format**

The first line contains a single integers N denoting the size of the array. The next line contains N space separated integers denoting the contents of the array.

**Constraints**

1<=N<=10^6 1<=A[I]<=10^9

**Output Format**

Print N space separated integers, i.e. the final sorted array.

**Sample Input 0**

5

4 3 1 5 2

**Sample Output 0**

1 2 3 4 5

1. WAP to sort an array of given numbers using Heap sort method.

Heaps can be used in sorting an array. In max-heaps, maximum element will always be at the root. Heap Sort uses this property of heap to sort the array.

**Input Format**

Array is in unsorted manner

**Constraints**

1<=T<10^5 1<=X<=10^6

**Output Format**

Output will come in a sorted manner

**Sample Input 0**

5

4

3

2

1

**Sample Output 0**

1 2 3 4 5

1. WAP to sort an array of given numbers Merge Sort method.

Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.

Idea:

Divide the unsorted list into N sublists, each containing 1 element.

Take adjacent pairs of two singleton lists and merge them to form a list of 2 elements.N will now convert into N/2 lists of size 2.

Repeat the process till a single sorted list of obtained.

While comparing two sublists for merging, the first element of both lists is taken into consideration. While sorting in ascending order, the element that is of a lesser value becomes a new element of the sorted list. This procedure is repeated until both the smaller sublists are empty and the new combined sublist comprises all the elements of both the sublists.

**Input Format**

First line contains one integer, N, size of array. Second line contains N space separated integers denoting the elements of the array A.

**Constraints**

1<=N<=10^6 1<=A[i]<=10^6

**Output Format**

Output will be sorted order

**Sample Input 0**

5

1 4 3 2 5

**Sample Output 0**

1. 2 3 4 5
2. Write a program for **deque** in which perform insert and delete operation using function.
3. Insert
4. Delete from rear
5. Delete from front

**Input Format**

First line contains the Menu followed Input for given menu.

**Constraints**

NA

**Output Format**

The output based on the Menu

**Sample Input 0**

1

67

1

12

1

15

3

2

3

2

3

**Sample Output 0**

->67

->12

->15

1. Write down a C program to implement a priority queue.

i) Insert

ii) Delete from rear

iii) Display

**Input Format**

First line contains the Menu followed Input for given menu.

**Constraints**

NA

**Output Format**

The output based on the Menu

**Sample Input 0**

1

23

1

45

1

78

3

2

3

2

3

**Sample Output 0**

->23

->45

->78

1. Write down a C program to implement multiple stack.
   1. Push
   2. Pop
   3. Display

**Input Format**

Enter the Number of stacks

First line contains the Menu followed Input for given menu.

**Constraints**

NA

**Output Format**

Print the output based on the Menu

**Sample Input 0**

1

23

1

45

2

34

2

65

3

1

**Sample Output 0**

23, 45

1. WAP in C to add Polynomial Linked List

You are given two polynomial functions, and you need to add them. Addition of a polynomial with another is the sum of all the terms that you receive after adding each term of first polynomial with all terms of second polynomial.

For more clarity see explanation of the sample case.

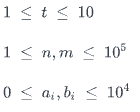
Both the polynomials ***a and b*** are given in coefficient form, that is , represents the coefficient of  , is   represents the coefficient of  , and so on.

**Input Format**

First line contains an integer *t* , the number of the test cases.

For each test case, there are three lines, first line contains two integers *n,* and *m*, the length of the two arrays. Second line for contains *n* integers and third line contains *m* integers, representing both the polynomials in coefficient form respectively.

**Constraints**



**Output Format**

**Output must contain *t* lines. Each line should contain *m+n* integers, denoting the coefficients of added polynomial.**

**Sample Input**

A: 5, 0, 10, 6

B: 1, 2, 4

**Sample Output**

sum : 6, 2, 14, 6

**Explanation**

The first input array represents "5 + 0x^1 + 10x^2 + 6x^3"

The second array represents "1 + 2x^1 + 4x^2"

And Output is "6 + 2x^1 + 14x^2 + 6x^3"

1. Given a string, reverse it using stack.

create an empty stack  
\*one by one push all characters of string to stack  
\*one by one pop all characters from stack and put them back to string.

**Input Format:**

**NA**

**Constraints:**

**NA**

**Output Format**

**Output must contain a reverse string**

**Sample Input**

**university**

**Sample Output**

**ytisrevinu**

1. Write a C program to Convert a postfix expression into infix expression.

**Input Format**

The first input will be a single integer N denoting the number of test cases to take. followed by N infix strings.

Binary operators (+, -, \*, / and ?) and parenthesis. Every integer, operator and parenthesis will be compulsorily separated by a SPACE. The symbol ‘?’ denotes the end of expression.

**Constraints**

NA

**Output Format**

N lines denoting postfix expression. Every integer and operator must be separated by a single SPACE.

**Sample Input**

ab\*c+

**Sample Output**

((a\*b)+c)

1. Write a program for stack representation using Linked List.
2. Push
3. Pop (Display "Stack is Empty" if stack is empty).
4. Display Stack Top (Display "Stack is Empty" if stack is empty).
5. Exit

**Input Format**

First line contains the Menu followed Input for given menu.

**Constraints**

NA

**Output Format**

Print the output based on the Menu

**Sample Input**

1

23

1

34

1

98

3

2

3

2

3

2

3

4

**Sample Output**

->23

->34

->98

Stack is Empty

1. Write a program for queue representation using Linked List.
2. Insert
3. Delete (Display "Queue is empty" if queue is empty).
4. Display (Display "NULL" if queue is empty).
5. Exit

**Input Format**

First line contains the Menu followed Input for given menu.

**Constraints**

NA

**Output Format**

Print the output based on the Menu

**Sample Input 1**

1

10

1

20

1

30

3

2

3

2

3

2

3

2

3

4

**Sample Output 1**

->10->20->30

->20->30

->30

NULL

Queue is empty

NULL

1. WAP to convert Postfix to Prefix

**Input Format**

The first input will be a single integer N denoting the number of test cases to take. followed by N infix strings.

Binary operators (+, -, \*, / and ?) and parenthesis. Every integer, operator and parenthesis will be compulsorily separated by a SPACE. The symbol ‘?’ denotes the end of expression.

**Constraints**

NA

**Output Format**

N lines denoting postfix expression. Every integer and operator must be separated by a single SPACE.

**Sample Input**

AB+CD-\*

**Sample Output**

\*+AB-CD