**Problem Reference Taken from website** practice.geeksforgeeks.org

**Problem 1**: Try this problem using ArrayList.

Implement different operations on an ArrayList A .

**Input:**

The first line of input contains an integer **T** denoting the no of test cases . Then **T** test cases follow. The first line of input contains an integer **Q** denoting the no of queries . Then in the next line are **Q** space separated queries .

A query can be of five types

1. a x (Adds an element x to the ArrayList A at the end )

2. b (Sorts the ArrayList A in ascending order )

3. c (Reverses the ArrayList A)

4. d (prints the size of the ArrayList)

5. e (prints space separated values of the ArrayList)

5. f (Sorts the ArrayList A in descending order)

**Output:**

The output for each test case will be space separated integers denoting the results of each query .

**Constraints:**

1<=T<=100

1<=Q<=100

**Example:**

**Input**

2

6

a 4 a 6 a 7 b c e

4

a 55 a 11 d e

**Output**

7 6 4

2 55 11

**Explanation :**

**For the first test case**

There are six queries. Queries are performed in this order

1. a 4 { ArrayList has 4 }

2. a 7 {ArrayList has 7 }

3. a 6 {ArrayList has 6}

4. b {sorts the ArrayList in ascending order, ArrayList now is 4 6 7}

5. c {reverse the ArrayList}

6. e {prints the element of the ArrayList 7 6 4}

**For the second test case**

There are four queries. Queries are performed in this order

1. a 55 (ArrayList A has 55}

2. a 11 (ArrayList A has 55 ,11}

3. d (prints the size of the ArrayList A ie. 2 )

4. e (prints the elements of the ArrayList A ie 55 11)

============================

**Problem 2:**

Given a ArrayList of **n** elements and a integer **q** defining the query number(which will be either 1 or 2) :

**Query 1** includes two integers **p** and **r**. Which means add a value r at and index p in the ArrayList and print the whole updated ArrayList.

**Query 2** includes one integer **p**. In this query print the index at which the value p is last found in the ArrayList. If the value p is not found in the ArrayList then print "**-1**".

**NOTE: Assume 0 based indexing**

**Input:**

First line of the input file contains an integer T, denoting the number of test cases. Then T test case follows. First line of each test case contains two integers n and q(space separated). Second Line of each test case consists of n spaced integers. Third line consists of 2 integers if the value of q is 1 otherwise consists of a single integer.

**Output:**

For each test case, print required output.

**Constraints:**

1<=T<=10

1<=n<=104

1<=element<=103

**Example:**

**Input:**

2

5 1

1 4 5 9 3

2 6

4 2

1 9 2 4

4

**Output:**

1 4 6 5 9 3

3

============================

**Problem 3:** Try Using LinkedList

Given only a pointer to a node to be deleted in a singly linked list. Print the whole Linked List after deletion.

**Input:**

The first line of input contains an element T, denoting the no of test cases. Then T test cases follow. Each test case contains two lines. The first line of each test case contains two integers N denoting the no of elements of the linked list and P denoting the number of the node which is to be deleted (starting from 0) . Then in the next line are N space separated values of the linked list.

**Output:**

The output for each test case will be the space separated elements of the updated linked list.

**Constraints:**

1<=T<=100

2<=N<=103

0<=P<(N-1)

**Example(To be used only for expected output):**

**Input:**

2

5 2

1 2 3 4 5

4 0

1 3 6 9

**Output:**

1 2 4 5

3 6 9

===============

**Problem 4: Try Using Set**

Implement different operations on a set s.

**Input:**

The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. The first line of input contains an integer Q denoting the no of queries . Then in the next line are Q space separated queries .

A query can be of four types

1. a x (inserts an element x to the set s)

2. b (prints the contents of the set s in increasing order)

3. c x (erases an element x from the set s)

4. d x (prints 1 if the element x is present in the set else print -1)

5. e (prints the size of the set s)

**Output:**

The output for each test case will be space separated integers denoting the results of each query .

**Constraints:**

1 <= T <= 100

1 <= Q <= 100

**Example:**

**Input:**

2

6

a 1 a 2 a 3 b c 2 b

5

a 1 a 5 e d 5 d 2

**Output:**

1 2 3 1 3

2 1 -1

**Explanation :**

**Testcase 1:**

There are six queries. Queries are performed in this order

1. a 1 { insert 1 to set now set has {1} }

2. a 2 {inserts 2 to set now set has {1,2} }

3. a 3 {inserts 3 to set now set has {1,2,3} }

4. b {prints the set contents ie 1,2,3}

5. c 2 {removes 2 from the set }

6. b {prints the set contents ie 1,3}

**Testcase 2:**

There are five queries. Queries are performed in this order

1. a 1 {inserts 1 to set now set has {1}}

2. a 11 {inserts 11 to set now set has {1,11}}

3. e {prints the size of the set ie 2}

4. d 5 {since five is present prints 1}

5. d 2 {since 2 is not present in the set prints -1}

**===================**

**Problem 5:**

Implement different operations on a LinkedHashset S.

**Input:**

The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. The first line of input contains an integer Q denoting the no of queries . Then in the next line are Q space separated queries .

A query can be of four types

1. a x (inserts an element x to the LinkedHashSet s)

2. b (prints the contents of the LinkedHashSet s in increasing order)

3. c x (erases an element x from the LinkedHashSet s)

4. d x (prints 1 if the element x is present in the LinkedHashSet else print -1)

5. e (prints the size of the LinkedHashSet s)

6. f (prints the contents of the LinkedHashSet s in insertion order)

**Output:**

The output for each test case will be space separated integers denoting the results of each query .

**Constraints:**

1<=T<=100

1<=Q<=100

**Example:**

**Input**

2

7

a 1 a 2 a 3 b c 2 b f

10

a 5 a 4 a 3 a 2 a 1 b c 1 c 3 b f

**Output**

1 2 3 1 3 1 3

1 2 3 4 5 2 4 5 5 4 2

**Explanation :**

**For the first test case:**

There are seven queries. Queries are performed in this order

1. a 1 { insert 1 to LinkedHashSet now set has {1} }

2. a 2 {inserts 2 to LinkedHashSet now set has {1,2} }

3. a 3 {inserts 3 to set now LinkedHashSet has {1,2,3} }

4. b {prints the LinkedHashSet contents ie 1,2,3}

5. c 2 {removes 2 from the LinkedHashSet }

6. b {prints the LinkedHashSet contents ie1,3}

7. f {prints the LinkedHashSet contents in inserted order ie 1,3}

**For the second test case:**

There are ten queries. Queries are performed in this order

1. a 5 {inserts 5 to set now set has {5} }

2. a 4 {inserts 4 to set now set has {5,4} }

3. a 3 {inserts 3 to set now set has {5,4,3} }

4. a 2 {inserts 2 to set now set has {5,4,3,2} }

5. a 1 {inserts 1 to set now set has {5,4,3,2,1} }

6. b {prints the LinkedHashSet contents in increasing order ie 1,2,3,4,5}

7. c 1 {removes 1 from the LinkedHashSet }

8. c 3 {removes 3 from the LinkedHashSet }

9. b {prints the LinkedHashSet contents in increasing order ie 2,4,5}

10. f {prints the LinkedHashSet contents in inserted order ie 5,4,2}

=============

**Problem 6:**

**Implement using TreeSet**

Given n strings added in a TreeSet and four characters a,b,c,d. Firstly print all the strings which are less than a , then print all the strings which are greater than or equal to b. In the end print all the strings which are greater than or equal to c and less than d i.e between c and d ( excluding d ).

**Input:**

The first line of input is T denoting the number of testcases. First line consists of an integer n. Next line contains n spaced strings. Last line of each test case consists of 4 space separated characters.

**Output:**

For each test case output will be in three lines:

First Line : Print all the strings which are less than a (char variable).

Second Line : Print all the strings which are greater than or equal to b (char variable).

Third Line : Print all the strings which are greater than or equal to c and less than d i.e between c and d ( excluding d ).

**Constraints:**

1<=T<=100

1<=N<=100

1<=Length of string<=100

**Example:**

**Input:**

1

3

sa ka da

s k c s

**Output:**

[da, ka]

[ka, sa]

[da, ka]

**==================**

**Problem 7:**

**Try this problem using HashMap.**

Given n key - value pairs and a seperate key in next line. If this key is found in the previous n keys then print its value otherwise print "-1".

**Input:**

First line of input contains a single integer T which denotes the number of test cases. T test cases follows, first line of each test case contains a integer n. Second line of each test case consists of n key(String)-value(Integer) pairs space separated. Third and last line of each test case consists of a single key(String).

**Output:**

If the key is found in the HashMap print its value otherwise print "-1".

**Constraints:**

1<=T<=100

1<=n,|Length of key|,value<=1000

**Example:**

**Input:**

2

3

sak 5 varun 7 vijay 3

sak

4

csdb 4 dsh 5 askj 8 adfs 9

dptu

**Output:**

5

-1

**======================**

**Problem 8:**

Implement different operations on Hashmap.

**Input:**

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. The first line of input contains an integer Q denoting the no of queries . Then in the next line are Q space separated queries .

A query can be of four types

1. a x y (adds an entry with key x and value y to the Hashmap)

2. b x (print value of x if present in the Hashmap else print -1. )

3. c (prints the size of the Hashmap)

4. d x (removes an entry with key x from the Hashmap)

**Output:**

The output for each test case will be space separated integers denoting the results of each query .

**Constraints:**

1<=T<=100

1<=Q<=100

**Example(To be used only for expected output):**

**Input**

2

5

a 1 2 a 66 3 b 66 d 1 c

3

a 1 66 b 5 c

**Output**

3 1

-1 1

**Explanation :**

For the first test case

There are five queries. Queries are performed in this order

1. a 1 2 --> map has a key 1 with value 2

2. a 66 3 ---> map has a key 66 with value 3

3. b 66 ---> prints the value of key 66 if its present in the map ie 3.

4. d 1 ---> removes an entry from map with key 1

5. c ---> prints the size of the map ie 1

**For the second test case**

There are three queries. Queries are performed in this order

1. a 1 66 ---> adds a key 1 with a value of 66 in the map

2. b 5 ---> since the key 5 is not present in the map hence -1 is printed.

3. c ---> prints the size of the map ie 1

**=================**

**Problem 9:**

Implement different operations on Treemap.

**Input:**

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. The first line of input contains an integer Q denoting the no of queries . Then in the next line are Q space separated queries .

A query can be of four types

1. a x y (adds an entry with key x and value y to the Treemap)

2. b x (print value of x if present in the Treemap else print -1. )

3. c (prints the size of the Treemap)

4. d x ( removes an entry with key x from the map )

5. e (print map sorted by key)

**Output:**

The output for each test case will be space separated integers denoting the results of each query.

**Constraints:**

1<=T<=100

1<=Q<=100

**Example(To be used only for expected output):**

**Input**

2

6

a 1 2 a 66 3 b 66 d 1 c e

4

a 1 66 b 5 e c

**Output**

3 1 66

-1 1 1

**Explanation :**

**For the first test case**

There are five queries. Queries are performed in this order

1. a 1 2 --> map has a key 1 with value 2

2. a 66 3 ---> map has a key 66 with value 3

3. b 66 ---> prints the value of key 66 if its present in the map ie 3.

4. d 1 ---> removes an entry from map with key 1

5. c ---> prints the size of the map ie 1

6. e ---> prints the map sorted by key

**For the second test case**

There are three queries. Queries are performed in this order

1. a 1 66 ---> adds a key 1 with a value of 66 in the map

2. b 5 ---> since the key 5 is not present in the map hence -1 is printed.

3. c ---> prints the size of the map ie 1

4. e ---> prints the map sorted by key

**================**

**Problem 10:**

Try this problem using Stack.

Given n elements of a stack , delete the middle element of the stack without using any additional data structure.

**Input:**

First line of input contains a single integer T which denotes the number of test cases. T test cases follows, first line of each test case contains a integer n. Second line consists of n spaced integers.

**Output:** Print the elements of the stack after deleting the middle element.

**Constraints:**

1<=T<=100

1<= n <=103

1<=Element<=104

**Example:**

**Input:**

2

5

1 2 3 4 5

6

1 4 9 2 6 5

**Output:**

5 4 2 1

5 6 2 4 1

**==============**

**Problem 11:**

**Input:**

The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. The first line of input contains an integer Q denoting the no of queries . Then in the next line are Q space separated queries .

A query can be of four types

1. a x (Pushes an element x at the end of the queue q )

2. r (if queue is not empty pops the front element and prints it, else prints -1)

3. s (prints the size of the queue)

4. p (if queue is not empty prints the front element of the queue, else prints -1)

**Output:**

The output for each test case will be space separated integers denoting the results of each query.

**Constraints:**

1<=T<=100

1<=Q<=100

**Example:**

**Input**

2

5

a 5 a 10 a 15 r p

4

a 5 r p s

**Output:**

5 10

5 -1 0

**EXPLANATION:**

In test case 1: In first query 5 is added to the queue.In second query 10 is added to the queue. In third query 15 is added to the queue. In fourth query 5 is removed from the queue and printed.In last query front element of the queue is printed i.e. 10.