Java Collection Assignment

1. Java Collection: ArrayList Exercises
Write a Java program to create a new array list, add some Movie names (string) and print out the collection.
Write a Java program to insert an element into the array list at the first position.
Write a Java program to retrieve an element (at a specified index) from a given array list.
Write a Java program to update specific array elements by given element.
Write a Java program to remove the third element from an array list.
Write a Java program to search for an element in an array list.
Write a Java program to sort a given array list.
Write a Java program to reverse elements in an array list.
Write a Java program to empty an array list.

```java
import java.util.*;

public class ArrayListDemo {

    static void displayList(List<?> list) {
        for (Object obj : list) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        ArrayList<String> movies = new ArrayList<>();

        movies.add("Gumnaam");
        movies.add("My Rainy Days");
        movies.add("Chalti Ka Naam Gaadi");
        movies.add("Madhumati");
        movies.add("A Moment to Remember");
        movies.add("Nezha");
        movies.add("Nezha 2");

        // print the collection
        System.out.println("Printing list--");
        for (String s : movies) {
            System.out.println(s);
```

```
        }

        System.out.println("\nAfter insert at first position--");
        movies.add(0, "Mr. and Mrs. '55");
        displayList(movies);

        System.out.println("\nElement at index 5--");
        System.out.println(movies.get(5));

        System.out.println("\nUpdate element at index 7--");
        movies.set(7, "Nezha: The Demon Child Conquers the Dragon
King");
        displayList(movies);

        System.out.println("\nRemove element at index 3--");
        movies.remove(3);
        displayList(movies);

        System.out.println("\nCheck if 'Gumnaam'' in list or not--");
        boolean b = movies.contains("Gumnaam");
        System.out.println(b);

        System.out.println("\nSort list--");
        Collections.sort(movies);
        displayList(movies);

        System.out.println("\nReverse list--");
        Collections.sort(movies, Collections.reverseOrder());
        displayList(movies);

        System.out.println("\nEmpty the list--");
        movies.clear();
        displayList(movies);
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac ArrayListDemo.java
PS D:\CDAC\Java\CollectionAssignment> java ArrayListDemo
Printing list--
Gumnaam
My Rainy Days
Chalti Ka Naam Gaadi
Madhumati
A Moment to Remember
Nezha
Nezha 2

After insert at first position--
Mr. and Mrs. '55
Gumnaam
My Rainy Days
Chalti Ka Naam Gaadi
Madhumati
A Moment to Remember
Nezha
Nezha 2

Element at index 5--
A Moment to Remember
```

```
Update element at index 7--
Mr. and Mrs. '55
Gumnaam
My Rainy Days
Chalti Ka Naam Gaadi
Madhumati
A Moment to Remember
Nezha
Nezha: The Demon Child Conquers the Dragon King

Remove element at index 3--
Mr. and Mrs. '55
Gumnaam
My Rainy Days
Madhumati
A Moment to Remember
Nezha
Nezha: The Demon Child Conquers the Dragon King

Check if 'Gumnaam'' in list or not--
true
```

```
Sort list--
A Moment to Remember
Gumnaam
Madhumati
Mr. and Mrs. '55
My Rainy Days
Nezha
Nezha: The Demon Child Conquers the Dragon King

Reverse list--
Nezha: The Demon Child Conquers the Dragon King
Nezha
My Rainy Days
Mr. and Mrs. '55
Madhumati
Gumnaam
A Moment to Remember

Empty the list--
```

2. Java Collection: LinkedList

Write a Java program to append the specified element to the end of a linked list of names.

Write a Java program to iterate through all elements in a linked list starting at the specified position.

Write a Java program to iterate a linked list in reverse order.

Write a Java program to insert the specified element at the specified position in the linked list.

Write a Java program to insert elements into the linked list at the first and last position.

Write a Java program to insert the specified element at the front of a linked list.

Write a Java program to insert some elements at the specified position into a linked list.

Write a Java program to get the first and last occurrence of the specified elements in a linked list.

Write a Java program to remove the first and last element from a linked list.

Write a Java program to swap two elements in a linked list.

Write a Java program to join two linked lists.

Write a Java program to check if a particular element exists in a linked list.

Write a Java program to convert a linked list to an array list.

Write a Java program to compare two linked lists.

Write a Java program to test whether a linked list is empty or not.

Write a Java program to replace an element in a linked list.

```java
import java.util.*;

public class LinkedListDemo {

    static void displayList(List<?> list) {
        for (Object obj : list) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        LinkedList<Integer> myLinkedList = new LinkedList<>();

        myLinkedList.add(111);
        myLinkedList.add(23);
        myLinkedList.add(3);
        myLinkedList.add(245);
        myLinkedList.add(111);
        myLinkedList.add(9);

        System.out.println("Linked List--");
        displayList(myLinkedList);

        System.out.println("\nElement added to last--");
        myLinkedList.addLast(101);
        displayList(myLinkedList);

        System.out.println("\nList iterating from index 3--");
        myLinkedList.listIterator(3);
        displayList(myLinkedList);

        System.out.println("\nReverse List--");
        Collections.sort(myLinkedList, Collections.reverseOrder());
        displayList(myLinkedList);

        System.out.println("\nAdd 15 at index 2--");
        myLinkedList.add(2, 15);
        displayList(myLinkedList);

        System.out.println("\nAdd element at first--");
```

```
        myLinkedList.addFirst(24);
        displayList(myLinkedList);

        System.out.println("\nAdd element at last--");
        myLinkedList.addLast(12);
        displayList(myLinkedList);

        System.out.println("\nAdd specified element at first--");
        myLinkedList.addFirst(84);
        displayList(myLinkedList);

        System.out.println("\nAdd 303 at index 6--");
        myLinkedList.add(6, 303);
        displayList(myLinkedList);

        System.out.println("\nFirst and Last occurrence of 111--");
        System.out.println("\nFirst occurrence of 111: " +
myLinkedList.indexOf(111));
        System.out.println("\nFirst occurrence of 111: " +
myLinkedList.lastIndexOf(111));

        System.out.println("\nRemove first element--");
        myLinkedList.removeFirst();
        displayList(myLinkedList);

        System.out.println("\nRemove last element--");
        myLinkedList.removeLast();
        displayList(myLinkedList);

        System.out.println("\nSwap elements at index 5 and 8--");
        Collections.swap(myLinkedList, 5, 8);
        displayList(myLinkedList);

        LinkedList<Integer> myLinkedList2 = new LinkedList<>();

        System.out.println("\nLinked List 2--");
        myLinkedList2.add(1110);
        myLinkedList2.add(2045);
        myLinkedList2.add(11001);
        myLinkedList2.add(19);
```

```
        displayList(myLinkedList2);

        System.out.println("\nJoin two linked list--");
        myLinkedList.addAll(myLinkedList2);
        displayList(myLinkedList);

        System.out.println("\nCheck list contains 245--");
        boolean b = myLinkedList.contains(245);
        System.out.println(b);

        System.out.println("\nConvert to Array--");
        Object[] obj = myLinkedList.toArray();
        for (Object o : obj) {
            System.out.println(o);
        }

        System.out.println("\nCompare two linked lists--");
        boolean b1 = myLinkedList.equals(myLinkedList2);
        System.out.println(b1);

        System.out.println("\nCheck whether empty--");
        System.out.println(myLinkedList.isEmpty());

        System.out.println("\nReplace an element--");
        myLinkedList.set(8, 1510);
        displayList(myLinkedList);
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac LinkedListDemo.java
PS D:\CDAC\Java\CollectionAssignment> java LinkedListDemo
Linked List--
111
23
3
245
111
9
```

```
Element added to last--
111
23
3
245
111
9
101

List iterating from index 3--
111
23
3
245
111
9
101

Reverse List--
245
111
111
101
23
9
3
```

```
Add 15 at index 2--
245
111
15
111
101
23
9
3

Add element at first--
24
245
111
15
111
101
23
9
3
```

```
Add element at last--
24
245
111
15
111
101
23
9
3
12

Add specified element at first--
84
24
245
111
15
111
101
23
9
3
12
```

```
Add 303 at index 6--
84
24
245
111
15
111
303
101
23
9
3
12

First and Last occurrence of 111--

First occurrence of 111: 3

First occurrence of 111: 5

Remove first element--
24
245
111
15
111
303
101
23
9
3
12
```

```
Remove last element--
24
245
111
15
111
303
101
23
9
3

Swap elements at index 5 and 8--
24
245
111
15
111
9
101
23
303
3

Linked List 2--
1110
2045
11001
19
```

```
Join two linked list--
24
245
111
15
111
9
101
23
303
3
1110
2045
11001
19

Check list contains 245--
true
```

```
Convert to Array--
24
245
111
15
111
9
101
23
303
3
1110
2045
11001
19
```

```
Compare two linked lists--
false

Check whether empty--
false

Replace an element--
24
245
111
15
111
9
101
23
1510
3
1110
2045
11001
19
```

3. Java Collection: HashSet Exercises

Write a Java program to append the specified element to the end of a hash set for Employee Id and Employee name.

Write a Java program to get the number of elements in a hash set.

Write a Java program to convert a hash set to an array.

Write a Java program to convert a hash set to a tree set.

Write a Java program to convert a hash set to a List/ArrayList.

Write a Java program to remove all of the elements from a hash set.

```java
import java.util.*;

class Employee implements Comparable<Employee> {
    int id;
    String name;

    Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }

    public int compareTo(Employee e) {
```

```
        int m = this.id - e.id;
        if (m > 0) {
            return 1;
        } else if (m < 0) {
            return -1;
        } else {
            return 0;
        }
    }

    public String toString() {
        return "ID: " + id + ", Name: " + name;
    }
}

public class HashSetDemo {

    static void displayHashSet(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        HashSet<Employee> hs = new HashSet<>();

        hs.add(new Employee(101, "Alice"));
        hs.add(new Employee(122, "Bob"));
        hs.add(new Employee(113, "Charles"));
        hs.add(new Employee(425, "David"));
        hs.add(new Employee(005, "Evan"));

        System.out.println("Hash Set--");
        displayHashSet(hs);

        System.out.println("\nNumber of elements--");
        System.out.println(hs.size());

        System.out.println("\nTo Array--");
        hs.toArray();
```

```
        displayHashSet(hs);

        System.out.println("\nTo TreeSet--");
        TreeSet<Employee> hashTree = new TreeSet<>(hs);
        displayHashSet(hashTree);

        System.out.println("\nTo List/ArrayList--");
        ArrayList<Employee> hashArr = new ArrayList<>();
        hashArr.addAll(hs);
        displayHashSet(hashArr);

        System.out.println("\nRemove all elements--");
        hs.removeAll(hs);
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac HashSetDemo.java
PS D:\CDAC\Java\CollectionAssignment> java HashSetDemo
Hash Set--
ID: 101, Name: Alice
ID: 425, Name: David
ID: 122, Name: Bob
ID: 5, Name: Evan
ID: 113, Name: Charles

Number of elements--
5

To Array--
ID: 101, Name: Alice
ID: 425, Name: David
ID: 122, Name: Bob
ID: 5, Name: Evan
ID: 113, Name: Charles
```

```
To TreeSet--
ID: 5, Name: Evan
ID: 101, Name: Alice
ID: 113, Name: Charles
ID: 122, Name: Bob
ID: 425, Name: David
```

```
To List/ArrayList--
ID: 101, Name: Alice
ID: 425, Name: David
ID: 122, Name: Bob
ID: 5, Name: Evan
ID: 113, Name: Charles

Remove all elements--
```

4. Java Collection: TreeSet

Write a Java program to create a new tree set, add some fruits (string) and print out the tree set.

Write a Java program to iterate through all elements in a tree set.

Write a Java program to add all the elements of a specified tree set to another tree set.

Write a Java program to create a reverse order view of the elements contained in a given tree set.

Write a Java program to find the numbers less than 7 in a tree set.

```
import java.util.*;

public class TreeSetDemo {

    static void displayTreeSet(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        TreeSet<String> fruits = new TreeSet<>();

        fruits.add("Apple");
        fruits.add("Custart Apple");
        fruits.add("Papaya");
        fruits.add("Guava");
        fruits.add("Sapota");
        fruits.add("Dates");

        System.out.println("Fruits Tree Set--");
        displayTreeSet(fruits);

        TreeSet<String> fruits2 = new TreeSet<>();
        fruits2.add("Orange");
        fruits2.add("Sweet Lime");
        fruits2.add("Dragon Fruit");
        fruits2.add("Banana");
        fruits2.add("Watermelon");
        fruits2.add("Grapes");
```

```
        System.out.println("\nFruits2 Tree Set--");
        displayTreeSet(fruits2);

        System.out.println("\nAdd elements of fruits2 to fruits--");
        fruits.addAll(fruits2);
        displayTreeSet(fruits);

        System.out.println("\nReverse Order View--");
        fruits.descendingSet();
        displayTreeSet(fruits);

        TreeSet<Integer> numberSet = new TreeSet<>();
        numberSet.add(10);
        numberSet.add(4);
        numberSet.add(8);
        numberSet.add(2);
        numberSet.add(12);
        numberSet.add(6);

        System.out.println("\nNumber Tree Set--");
        displayTreeSet(numberSet);

        System.out.println("\nNumbers less than 7--");
        System.out.println(numberSet.headSet(7));
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac TreeSetDemo.java
PS D:\CDAC\Java\CollectionAssignment> java TreeSetDemo
Fruits Tree Set--
Apple
Custart Apple
Dates
Guava
Papaya
Sapota

Fruits2 Tree Set--
Banana
Dragon Fruit
Grapes
Orange
Sweet Lime
Watermelon
```

```
Add elements of fruits2 to fruits--
Apple
Banana
Custart Apple
Dates
Dragon Fruit
Grapes
Guava
Orange
Papaya
Sapota
Sweet Lime
Watermelon
```

```
Reverse Order View--
Apple
Banana
Custart Apple
Dates
Dragon Fruit
Grapes
Guava
Orange
Papaya
Sapota
Sweet Lime
Watermelon
```

```
Number Tree Set--
2
4
6
8
10
12

Numbers less than 7--
[2, 4, 6]
```

5. Java Collection: HashMap
Write a Java program to associate the specified value with the specified key in a HashMap.
Write a Java program to count the number of key-value (size) mappings in a map.
Write a Java program to copy all of the mappings from the specified map to another map.
Write a Java program to remove all of the mappings from a map.
Write a Java program to test if a map contains a mapping for the specified key.
Write a Java program to test if a map contains a mapping for the specified value.

```
import java.util.*;

public class HashMapDemo {
    public static void main(String[] args) {
        HashMap<Integer, String> myHashMap = new HashMap<>();

        myHashMap.put(8, "S.Coups");
        myHashMap.put(4, "Jeonghan");
        myHashMap.put(30, "Joshua");
        myHashMap.put(10, "Jun");
        myHashMap.put(15, "Hoshi");

        System.out.println("Hash Map--");
        Set<Map.Entry<Integer, String>> e = myHashMap.entrySet();
        for (Map.Entry<Integer, String> entry : e) {
            System.out.println(entry.getKey() + ":" +
entry.getValue());
        }

        System.out.println("\nNumber of key-value map--");
        System.out.println(myHashMap.size());

        System.out.println("\nCopy myHashMap to myHashMapCopy--");
        HashMap<Integer, String> myHashMapCopy = new HashMap<>();
        myHashMapCopy.putAll(myHashMap);

        System.out.println("\nmyHashMapCopy--");
        for (Map.Entry<Integer, String> entry1 :
myHashMapCopy.entrySet()) {
```

```
            System.out.println(entry1.getKey() + " : " +
entry1.getValue());
        }

        System.out.println("\nRemove all from myHashMapCopy--");
        myHashMapCopy.clear();
        for (Map.Entry<Integer, String> entry1 :
myHashMapCopy.entrySet()) {
            System.out.println(entry1.getKey() + " : " +
entry1.getValue());
        }

        System.out.println("\nCheck if map contains specified key
(here, 12)--");
        System.out.println(myHashMap.containsKey(12));

        System.out.println("\nCheck if map contains specified value
(here, 'Jun')--");
        System.out.println(myHashMap.containsValue("Jun"));
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac HashMapDemo.java
PS D:\CDAC\Java\CollectionAssignment> java HashMapDemo
Hash Map--
4:Jeonghan
8:S.Coups
10:Jun
30:Joshua
15:Hoshi

Number of key-value map--
5
```

```
Copy myHashMap to myHashMapCopy--

myHashMapCopy--
8 : S.Coups
10 : Jun
4 : Jeonghan
30 : Joshua
15 : Hoshi

Remove all from myHashMapCopy--

Check if map contains specified key (here, 12)--
false

Check if map contains specified value (here, 'Jun')--
true
```

Practice Problem: Ex:1
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

a4 a6 a7bce
4
a 55 a 11 de
Output
764
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 5 6 7}
5. c {reverse the ArrayList}
6. e {prints the element of the ArrayList 7 6 4}
For the sec test case
There are four queries. Queries are performed in this order
1. a 55 (ArrayList A has 55}
(prints the size of the ArrayList A ie. 2)
2. a 11
(ArrayList A has 55,11}
3. d
4. e
(prints the elements of the ArrayList A ie 55 11)

```
import java.util.*;

public class Problem1 {

    static void displayList(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        ArrayList<Integer> list1 = new ArrayList<>();
        list1.add(12);
        list1.add(99);
        list1.add(520);
        list1.add(6);
```

```
        System.out.println("\nArray List--");
        displayList(list1);

        System.out.println("\nSorted Array List--");
        Collections.sort(list1);
        displayList(list1);

        System.out.println("\nReverse Array List--");
        Collections.sort(list1, Collections.reverseOrder());
        displayList(list1);

        System.out.println("\nArray List Size--");
        System.out.println(list1.size());
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac Problem1.java
PS D:\CDAC\Java\CollectionAssignment> java Problem1

Array List--
12
99
520
6

Sorted Array List--
6
12
99
520

Reverse Array List--
520
99
12
6

Array List Size--
4
```

Practice Problem: Ex:2
ArrayList are dynamic size arrays. Try this problem using ArrayList.
Given an ArrayList of N elements and an integer Q defining the type of query(which will be either 1 or 2): Q = 1 includes two integers p and r. Which means insert the value r at index p in the ArrayList and print the whole updated ArrayList.

Q = 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
Example 1:
Input:
N = 5, Q = 1
A[] = {1, 4, 5, 9, 3}
Query []
Output:
= {2,6}
1 4 6 5 9 3
Explanation:
p=Query [0]=2
r=Query [1]=6
After inserting the element r=6 at index p=2, the updated arraylist ={1,4,6,5,9,3}
Example 2:
Input:
N = 4, Q
= 2
A[]= {1, 9, 2, 4}
Query[] = {4}
Output:
Explanation:
3
p = 4
The element 4 is last found
in A at index = 3
Your Task:
You don't need to read input or print anything. Your task is to complete the function solve() which takes the N (number of elements in Array A),ArrayList A, Q(Type of the of query) and the ArrayList Query. If the Q = 1 then return the updated ArrayList of integers. else return the ArrayList which contains the index at which the value p is last found in the ArrayList A (where p = Query[0]),If the value of p is not found then return the ArrayList which contains -1.
Expected Time Complexity: O(N)
Expected Auxiliary Space: O(N) Constraints:
1 <= N <= 104
1 <=Q <= 2
If Q = 1 then size of Query is 2,

where Query[0] represents the value of p and Query[0] represents the value of r.
If Q = 2 then size of Query is 1,
where Query[0] represents the value of p.
1 <= A[i] <= 103

```
import java.util.*;

public class Problem2 {

    static void displayList(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        ArrayList<Integer> list1 = new ArrayList<>();

        list1.add(101);
        list1.add(15);
        list1.add(12);
        list1.add(520);
        list1.add(4);
        list1.add(17);
        list1.add(13);

        System.out.println("Array List--");
        displayList(list1);

        System.out.println("\nInsert 10 at index 3--");
        list1.add(3, 10);
        displayList(list1);

        System.out.println("\n50 is last found at index--");
        System.out.println(list1.lastIndexOf(50));
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac Problem2.java
PS D:\CDAC\Java\CollectionAssignment> java Problem2
Array List--
101
15
12
520
4
17
13

Insert 10 at index 3--
101
15
12
10
520
4
17
13

50 is last found at index--
-1
```

Practice Problem: Ex:3
Java provides an inbuilt object type called Stack. It is a collection that is based on the last in first out (LIFO) principle. Try this problem using Stack.
Given n elements of a stack st where the first value is the bottom-most value of the stack and the last one is the element at top of the stack, delete the middle element of the stack without using any additional data structure.
Example 1:
Input: n = 5
st = {1, 2, 3, 4, 5}
Output: 5 4 2 1
Explanation: The middle element is 3. If
it is deleted and then the values are seen from top, this will be the order.
Example 2:
Input: n = 6
st = {1, 4, 9, 2, 6, 5}
Output: 5 6 2 4 1
Explanation: The middle element is 9 and if
it is deleted this will be the stack traversal.
Your Task:

You do not need to read input or print anything. Your task is to complete the function deleteMid() which takes n and st as input parameters and returns a stack where the middle element is deleted.
Expected Time Complexity: O(n)
Expected Auxiliary Space: O(n)
Constraints:
2 ≤ n ≤ 103
1 ≤ st[i] ≤ 104

```
import java.util.*;

public class Problem3 {

    static void displayStack(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public static void main(String[] args) {
        Stack<Integer> s = new Stack<>();

        s.add(101);
        s.add(15);
        s.add(520);
        s.add(12);
        s.add(4);
        s.add(17);
        s.add(13);

        System.out.println("Stack--");
        displayStack(s);

        System.out.println("\nRemove middle element");
        int size = s.size();
        s.remove(size / 2);
        displayStack(s);
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac Problem3.java
PS D:\CDAC\Java\CollectionAssignment> java Problem3
Stack--
101
15
520
12
4
17
13

Remove middle element
101
15
520
4
17
13
```

Practice Problem: Ex:4
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. cx (erases an element x from the sets)
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 bc2b
5
a 1a5ed5d2

Output:
12313
21-1
Explanation:
Testcase 1:
There are six queries. Queries are performed in this order
1. a 1
2. a 2 3. a 3 4. b
5. c 2
6. b
{ insert 1 to set now set has {1}}
{inserts 2 to set now set has {1,2} } {inserts 3 to set now set has {1,2,3}} {prints the set contents ie 1,2,3}
{removes 2 from the set}
{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}

```
import java.util.*;

public class Problem4 implements Comparable<Integer> {

    static void displaySet(Collection<?> col) {
        for (Object obj : col) {
            System.out.println(obj);
        }
    }

    public int compareTo(Integer i) {
        if (i > 0) {
            return 1;
        } else if (i < 0) {
```

```
            return -1;
        } else {
            return 0;
        }
    }

    public static void main(String[] args) {
        Set<Integer> s = new TreeSet<>();

        s.add(101);
        s.add(15);
        s.add(520);
        s.add(12);
        s.add(4);
        s.add(17);
        s.add(13);

        System.out.println("Set--");
        displaySet(s);

        System.out.println("\nSet in increasing order--");
        displaySet(s);

        System.out.println("\nRemove 4 from set--");
        s.remove(4);
        displaySet(s);

        System.out.println("\nCheck if 17 in set--");
        int res = s.contains(17) ? 1 : -1;
        System.out.println(res);

        System.out.println("\nSize of set--");
        System.out.println(s.size());
    }
}
```

```
PS D:\CDAC\Java\CollectionAssignment> javac Problem4.java
PS D:\CDAC\Java\CollectionAssignment> java Problem4
Set--
4
12
13
15
17
101
520

Set in increasing order--
4
12
13
15
17
101
520

Remove 4 from set--
12
13
15
17
101
520

Check if 17 in set--
1

Size of set--
6
```