Java Collection Assignment

Java Collection: ArrayList Exercises

1. 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.

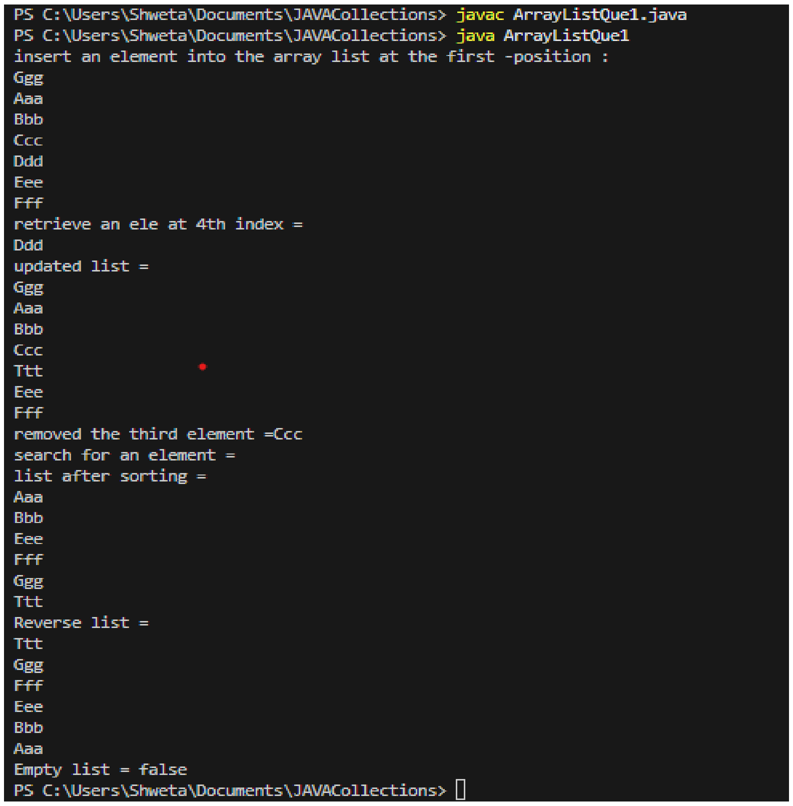
-Write a Java program to remove the third element from an array list.

-Write a Java program to search for an element in an array list.

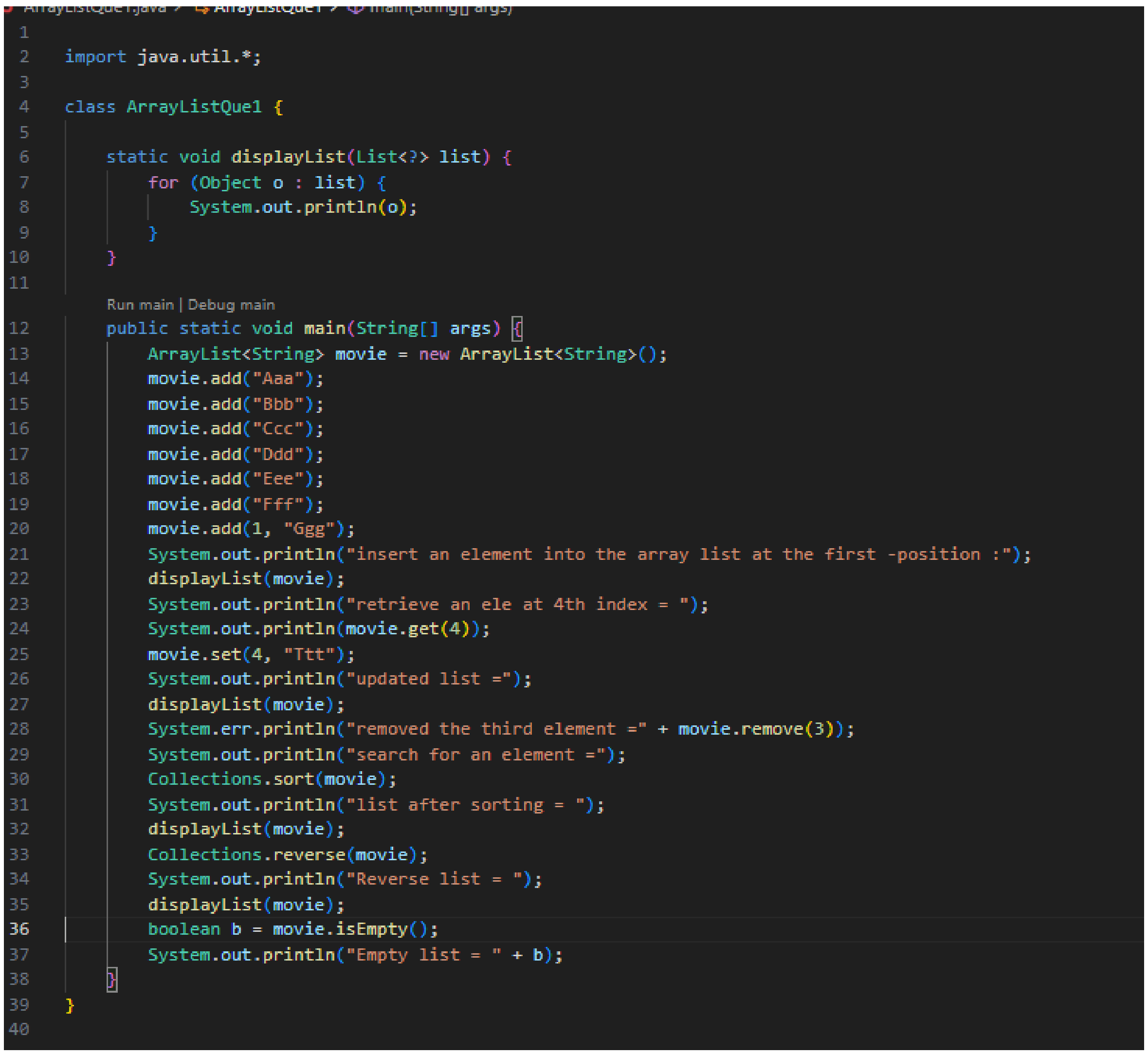
-Write a Java program to sort a given array list.

-Write a Java program to reverse elements in an array list.

-Write a Java program to empty an array list.



```
PS C:\Users\Shweta\Documents\JAVACollections> javac ArrayListQue1.java
PS C:\Users\Shweta\Documents\JAVACollections> java ArrayListQue1
insert an element into the array list at the first -position :
Ggg
Aaa
Bbb
Ccc
Ddd
Eee
Fff
retrieve an ele at 4th index =
Ddd
updated list =
Ggg
Aaa
Bbb
Ccc
Ttt
Eee
Fff
removed the third element =Ccc
search for an element =
list after sorting =
Aaa
Bbb
Eee
Fff
Ggg
Ttt
Reverse list =
Ttt
Ggg
Fff
Eee
Bbb
Aaa
Empty list = false
PS C:\Users\Shweta\Documents\JAVACollections>
```



```java
import java.util.*;

class ArrayListQue1 {

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

    Run main | Debug main
    public static void main(String[] args) {
        ArrayList<String> movie = new ArrayList<String>();
        movie.add("Aaa");
        movie.add("Bbb");
        movie.add("Ccc");
        movie.add("Ddd");
        movie.add("Eee");
        movie.add("Fff");
        movie.add(1, "Ggg");
        System.out.println("insert an element into the array list at the first -position :");
        displayList(movie);
        System.out.println("retrieve an ele at 4th index = ");
        System.out.println(movie.get(4));
        movie.set(4, "Ttt");
        System.out.println("updated list =");
        displayList(movie);
        System.err.println("removed the third element =" + movie.remove(3));
        System.out.println("search for an element =");
        Collections.sort(movie);
        System.out.println("list after sorting = ");
        displayList(movie);
        Collections.reverse(movie);
        System.out.println("Reverse list = ");
        displayList(movie);
        boolean b = movie.isEmpty();
        System.out.println("Empty list = " + b);
    }
}
```

Java Collection: LinkedList

1. 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.

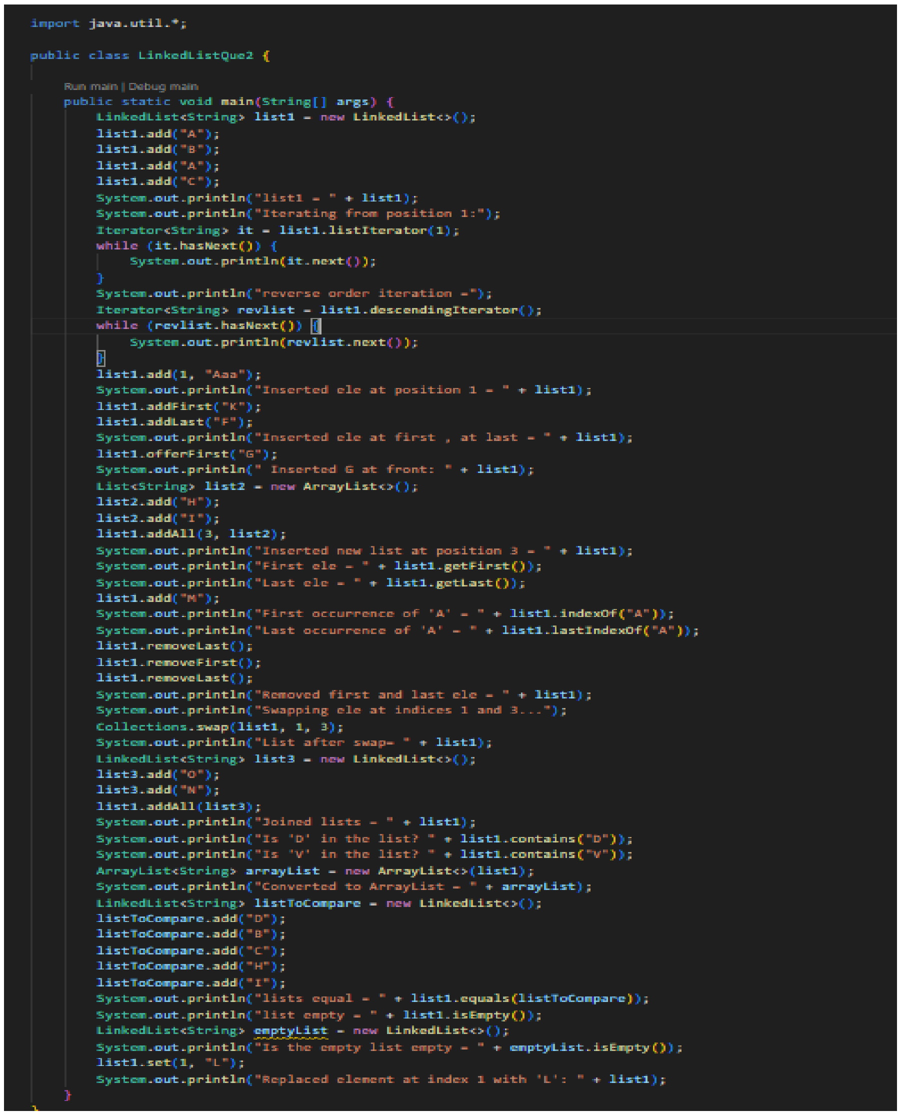
-Write a Java program to check if a particular element exists in a linked list.

-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.

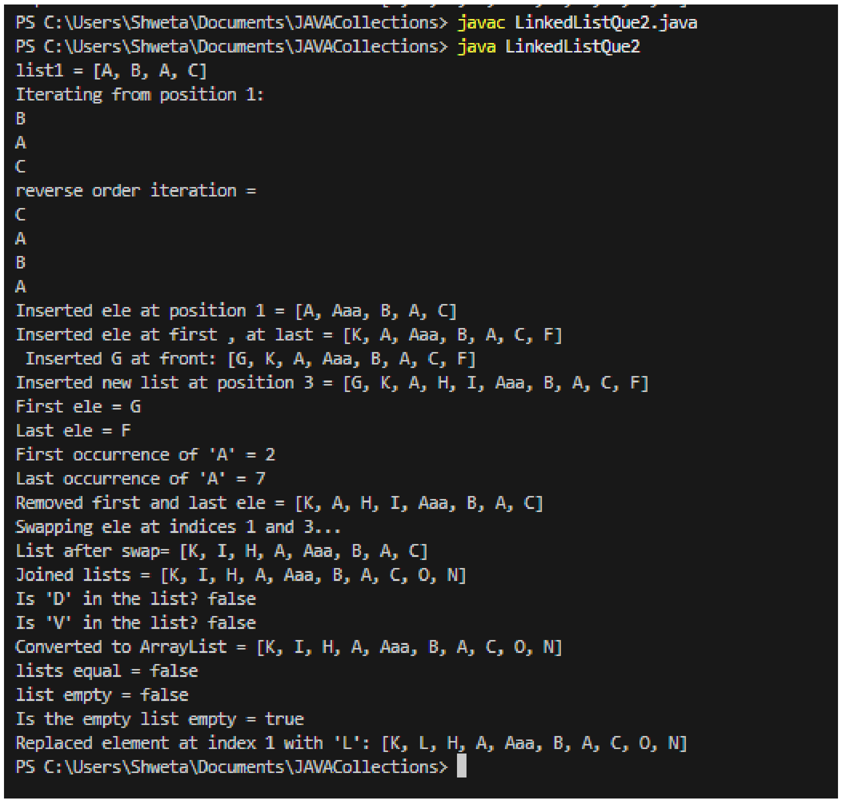
-Write a Java program to replace an element in a linked list.



```java
import java.util.*;

public class LinkedListQue2 {

    Run main | Debug main
    public static void main(String[] args) {
        LinkedList<String> list1 = new LinkedList<>();
        list1.add("A");
        list1.add("B");
        list1.add("A");
        list1.add("C");
        System.out.println("list1 = " + list1);
        System.out.println("Iterating from position 1:");
        Iterator<String> it = list1.listIterator(1);
        while (it.hasNext()) {
            System.out.println(it.next());
        }
        System.out.println("reverse order iteration =");
        Iterator<String> revlist = list1.descendingIterator();
        while (revlist.hasNext()) {
            System.out.println(revlist.next());
        }
        list1.add(1, "Aaa");
        System.out.println("Inserted ele at position 1 = " + list1);
        list1.addFirst("K");
        list1.addLast("F");
        System.out.println("Inserted ele at first , at last = " + list1);
        list1.offerFirst("G");
        System.out.println(" Inserted G at front: " + list1);
        List<String> list2 = new ArrayList<>();
        list2.add("H");
        list2.add("I");
        list1.addAll(3, list2);
        System.out.println("Inserted new list at position 3 = " + list1);
        System.out.println("First ele = " + list1.getFirst());
        System.out.println("Last ele = " + list1.getLast());
        list1.add("M");
        System.out.println("First occurrence of 'A' = " + list1.indexOf("A"));
        System.out.println("Last occurrence of 'A' = " + list1.lastIndexOf("A"));
        list1.removeLast();
        list1.removeFirst();
        list1.removeLast();
        System.out.println("Removed first and last ele = " + list1);
        System.out.println("Swapping ele at indices 1 and 3...");
        Collections.swap(list1, 1, 3);
        System.out.println("List after swap= " + list1);
        LinkedList<String> list3 = new LinkedList<>();
        list3.add("O");
        list3.add("N");
        list1.addAll(list3);
        System.out.println("Joined lists = " + list1);
        System.out.println("Is 'D' in the list? " + list1.contains("D"));
        System.out.println("Is 'V' in the list? " + list1.contains("V"));
        ArrayList<String> arrayList = new ArrayList<>(list1);
        System.out.println("Converted to ArrayList = " + arrayList);
        LinkedList<String> listToCompare = new LinkedList<>();
        listToCompare.add("D");
        listToCompare.add("B");
        listToCompare.add("C");
        listToCompare.add("H");
        listToCompare.add("I");
        System.out.println("lists equal = " + list1.equals(listToCompare));
        System.out.println("list empty = " + list1.isEmpty());
        LinkedList<String> emptyList = new LinkedList<>();
        System.out.println("Is the empty list empty = " + emptyList.isEmpty());
        list1.set(1, "L");
        System.out.println("Replaced element at index 1 with 'L': " + list1);
    }
}
```



```
PS C:\Users\Shweta\Documents\JAVACollections> javac LinkedListQue2.java
PS C:\Users\Shweta\Documents\JAVACollections> java LinkedListQue2
list1 = [A, B, A, C]
Iterating from position 1:
B
A
C
reverse order iteration =
C
A
B
A
Inserted ele at position 1 = [A, Aaa, B, A, C]
Inserted ele at first , at last = [K, A, Aaa, B, A, C, F]
 Inserted G at front: [G, K, A, Aaa, B, A, C, F]
Inserted new list at position 3 = [G, K, A, H, I, Aaa, B, A, C, F]
First ele = G
Last ele = F
First occurrence of 'A' = 2
Last occurrence of 'A' = 7
Removed first and last ele = [K, A, H, I, Aaa, B, A, C]
Swapping ele at indices 1 and 3...
List after swap= [K, I, H, A, Aaa, B, A, C]
Joined lists = [K, I, H, A, Aaa, B, A, C, O, N]
Is 'D' in the list? false
Is 'V' in the list? false
Converted to ArrayList = [K, I, H, A, Aaa, B, A, C, O, N]
lists equal = false
list empty = false
Is the empty list empty = true
Replaced element at index 1 with 'L': [K, L, H, A, Aaa, B, A, C, O, N]
PS C:\Users\Shweta\Documents\JAVACollections>
```

Java Collection: HashSet Exercises

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

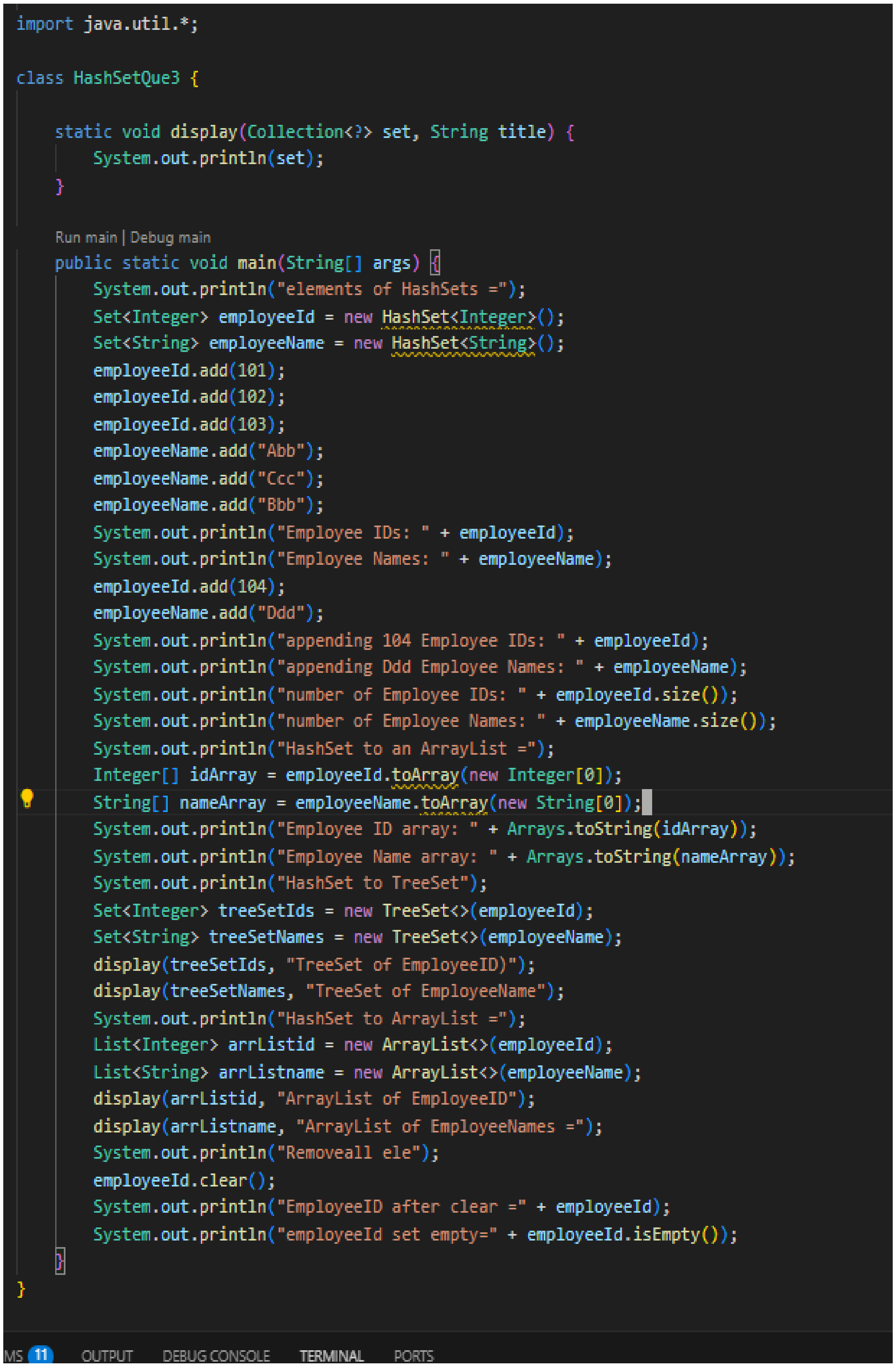
   Employee name.
   -Write a Java program to get the number of elements in a hash set.

   -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.

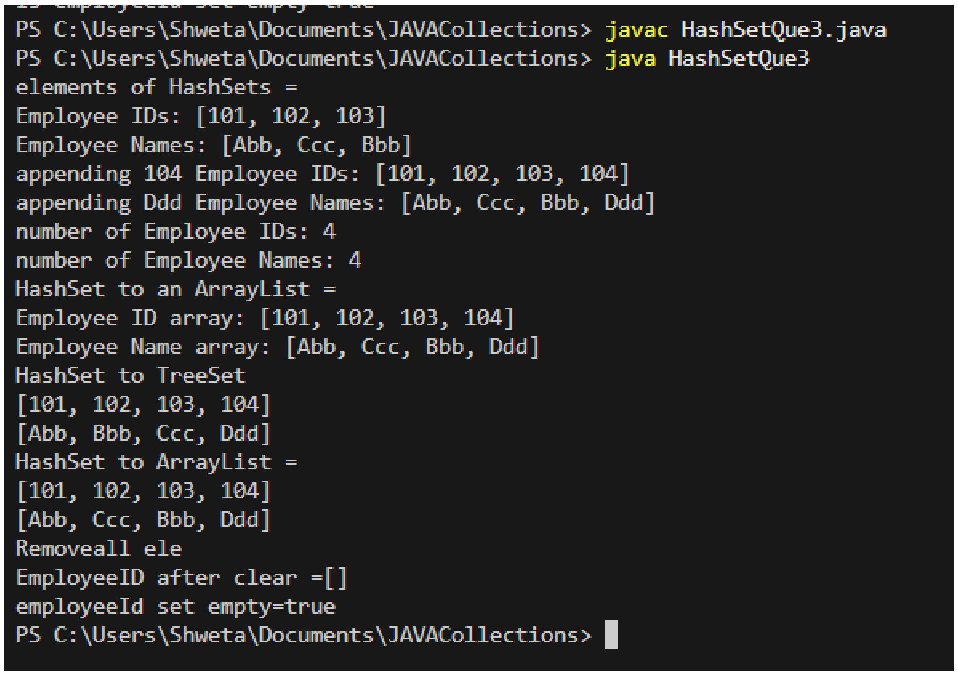


```java
import java.util.*;

class HashSetQue3 {

    static void display(Collection<?> set, String title) {
        System.out.println(set);
    }

    Run main | Debug main
    public static void main(String[] args) {
        System.out.println("elements of HashSets =");
        Set<Integer> employeeId = new HashSet<Integer>();
        Set<String> employeeName = new HashSet<String>();
        employeeId.add(101);
        employeeId.add(102);
        employeeId.add(103);
        employeeName.add("Abb");
        employeeName.add("Ccc");
        employeeName.add("Bbb");
        System.out.println("Employee IDs: " + employeeId);
        System.out.println("Employee Names: " + employeeName);
        employeeId.add(104);
        employeeName.add("Ddd");
        System.out.println("appending 104 Employee IDs: " + employeeId);
        System.out.println("appending Ddd Employee Names: " + employeeName);
        System.out.println("number of Employee IDs: " + employeeId.size());
        System.out.println("number of Employee Names: " + employeeName.size());
        System.out.println("HashSet to an ArrayList =");
        Integer[] idArray = employeeId.toArray(new Integer[0]);
        String[] nameArray = employeeName.toArray(new String[0]);
        System.out.println("Employee ID array: " + Arrays.toString(idArray));
        System.out.println("Employee Name array: " + Arrays.toString(nameArray));
        System.out.println("HashSet to TreeSet");
        Set<Integer> treeSetIds = new TreeSet<>(employeeId);
        Set<String> treeSetNames = new TreeSet<>(employeeName);
        display(treeSetIds, "TreeSet of EmployeeID)");
        display(treeSetNames, "TreeSet of EmployeeName");
        System.out.println("HashSet to ArrayList =");
        List<Integer> arrListid = new ArrayList<>(employeeId);
        List<String> arrListname = new ArrayList<>(employeeName);
        display(arrListid, "ArrayList of EmployeeID");
        display(arrListname, "ArrayList of EmployeeNames =");
        System.out.println("Removeall ele");
        employeeId.clear();
        System.out.println("EmployeeID after clear =" + employeeId);
        System.out.println("employeeId set empty=" + employeeId.isEmpty());
    }
}
```



```
PS C:\Users\Shweta\Documents\JAVACollections> javac HashSetQue3.java
PS C:\Users\Shweta\Documents\JAVACollections> java HashSetQue3
elements of HashSets =
Employee IDs: [101, 102, 103]
Employee Names: [Abb, Ccc, Bbb]
appending 104 Employee IDs: [101, 102, 103, 104]
appending Ddd Employee Names: [Abb, Ccc, Bbb, Ddd]
number of Employee IDs: 4
number of Employee Names: 4
HashSet to an ArrayList =
Employee ID array: [101, 102, 103, 104]
Employee Name array: [Abb, Ccc, Bbb, Ddd]
HashSet to TreeSet
[101, 102, 103, 104]
[Abb, Bbb, Ccc, Ddd]
HashSet to ArrayList =
[101, 102, 103, 104]
[Abb, Ccc, Bbb, Ddd]
Removeall ele
EmployeeID after clear =[]
employeeId set empty=true
PS C:\Users\Shweta\Documents\JAVACollections>
```

Java Collection: TreeSet

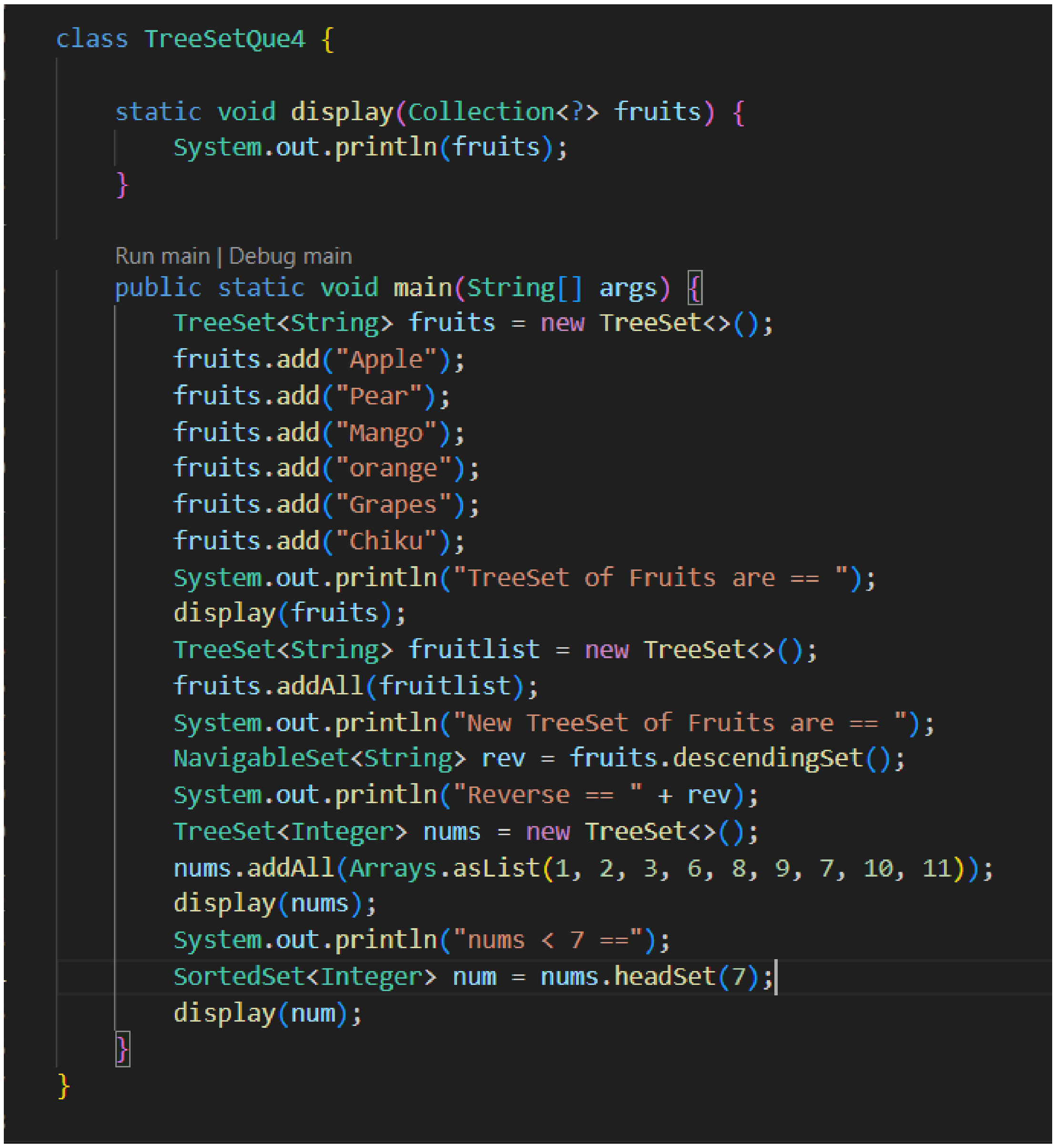
1. Write a Java program to create a new tree set, add some fruits (string) and print out the tree set.

   -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.

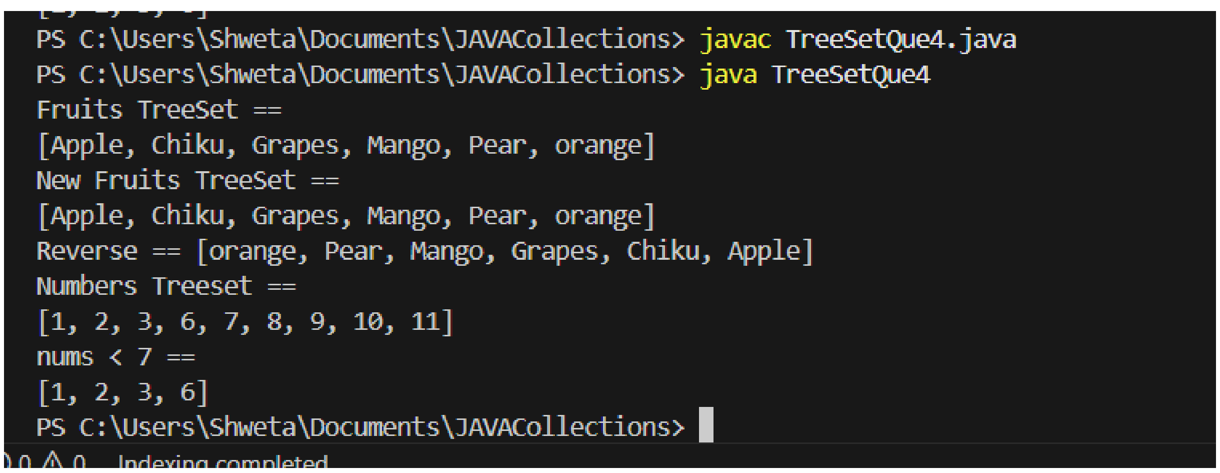
   -Write a Java program to find the numbers less than 7 in a tree set.



```java
class TreeSetQue4 {

    static void display(Collection<?> fruits) {
        System.out.println(fruits);
    }

    Run main | Debug main
    public static void main(String[] args) {
        TreeSet<String> fruits = new TreeSet<>();
        fruits.add("Apple");
        fruits.add("Pear");
        fruits.add("Mango");
        fruits.add("orange");
        fruits.add("Grapes");
        fruits.add("Chiku");
        System.out.println("TreeSet of Fruits are == ");
        display(fruits);
        TreeSet<String> fruitlist = new TreeSet<>();
        fruits.addAll(fruitlist);
        System.out.println("New TreeSet of Fruits are == ");
        NavigableSet<String> rev = fruits.descendingSet();
        System.out.println("Reverse == " + rev);
        TreeSet<Integer> nums = new TreeSet<>();
        nums.addAll(Arrays.asList(1, 2, 3, 6, 8, 9, 7, 10, 11));
        display(nums);
        System.out.println("nums < 7 ==");
        SortedSet<Integer> num = nums.headSet(7);
        display(num);
    }
}
```



```
PS C:\Users\Shweta\Documents\JAVACollections> javac TreeSetQue4.java
PS C:\Users\Shweta\Documents\JAVACollections> java TreeSetQue4
Fruits TreeSet ==
[Apple, Chiku, Grapes, Mango, Pear, orange]
New Fruits TreeSet ==
[Apple, Chiku, Grapes, Mango, Pear, orange]
Reverse == [orange, Pear, Mango, Grapes, Chiku, Apple]
Numbers Treeset ==
[1, 2, 3, 6, 7, 8, 9, 10, 11]
nums < 7 ==
[1, 2, 3, 6]
PS C:\Users\Shweta\Documents\JAVACollections>
```

⊗ 0 ⚠ 0 Indexing completed

Java Collection: HashMap

1. Write a Java program to associate the specified value with the specified key in a HashMap.

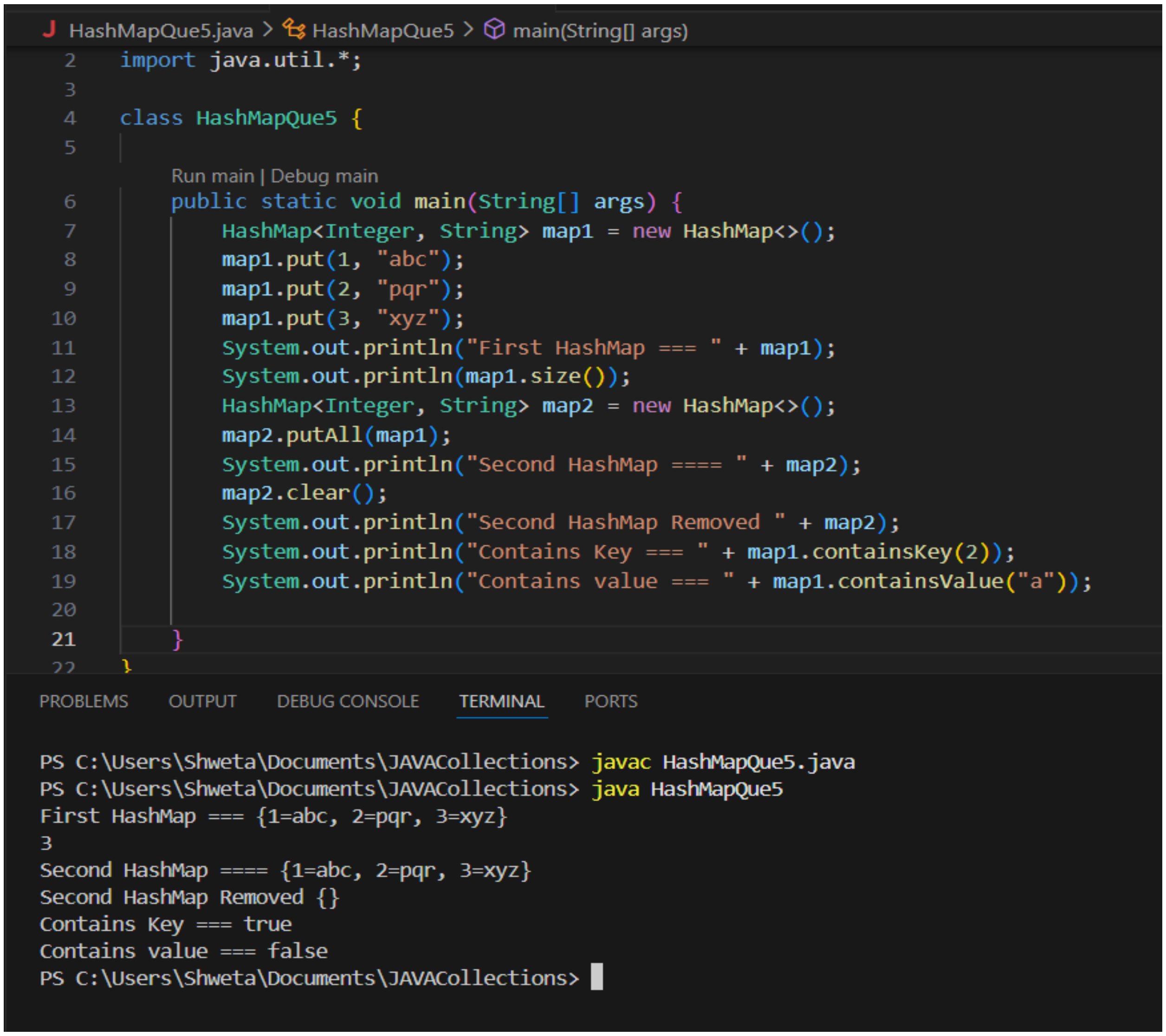
   -Write a Java program to count the number of key-value (size) mappings in a map.

   -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.



```java
import java.util.*;

class HashMapQue5 {

    Run main | Debug main
    public static void main(String[] args) {
        HashMap<Integer, String> map1 = new HashMap<>();
        map1.put(1, "abc");
        map1.put(2, "pqr");
        map1.put(3, "xyz");
        System.out.println("First HashMap === " + map1);
        System.out.println(map1.size());
        HashMap<Integer, String> map2 = new HashMap<>();
        map2.putAll(map1);
        System.out.println("Second HashMap ==== " + map2);
        map2.clear();
        System.out.println("Second HashMap Removed " + map2);
        System.out.println("Contains Key === " + map1.containsKey(2));
        System.out.println("Contains value === " + map1.containsValue("a"));

    }
}
```

```
PS C:\Users\Shweta\Documents\JAVACollections> javac HashMapQue5.java
PS C:\Users\Shweta\Documents\JAVACollections> java HashMapQue5
First HashMap === {1=abc, 2=pqr, 3=xyz}
3
Second HashMap ==== {1=abc, 2=pqr, 3=xyz}
Second HashMap Removed {}
Contains Key === true
Contains value === false
PS C:\Users\Shweta\Documents\JAVACollections>
```

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}

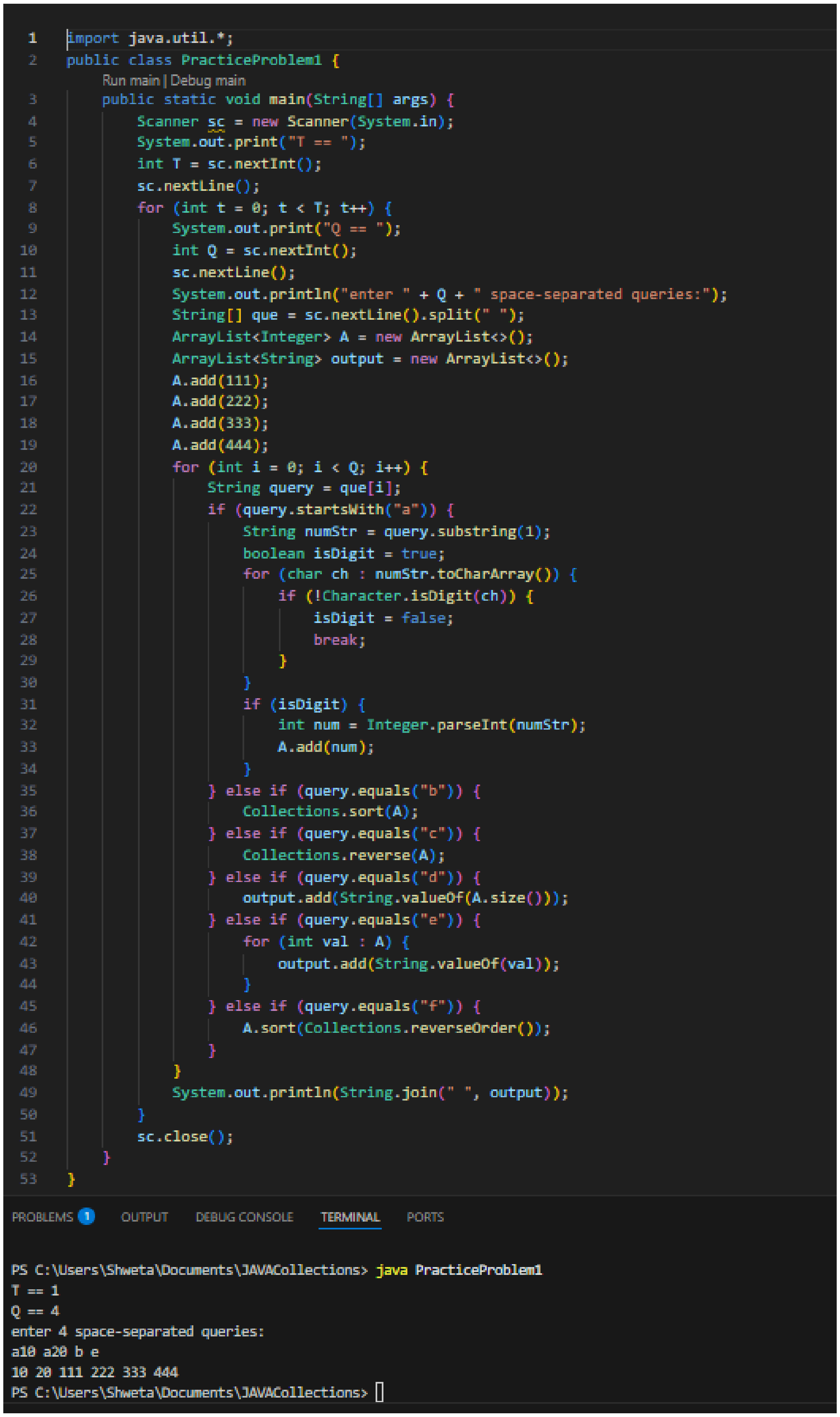
    (prints the size of the ArrayList A ie. 2)

2. a 11

    (ArrayList A has 55,11**}**

3. d
4. e

    (prints the elements of the ArrayList A ie 55 11)



```java
import java.util.*;
public class PracticeProblem1 {
    Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("T == ");
        int T = sc.nextInt();
        sc.nextLine();
        for (int t = 0; t < T; t++) {
            System.out.print("Q == ");
            int Q = sc.nextInt();
            sc.nextLine();
            System.out.println("enter " + Q + " space-separated queries:");
            String[] que = sc.nextLine().split(" ");
            ArrayList<Integer> A = new ArrayList<>();
            ArrayList<String> output = new ArrayList<>();
            A.add(111);
            A.add(222);
            A.add(333);
            A.add(444);
            for (int i = 0; i < Q; i++) {
                String query = que[i];
                if (query.startsWith("a")) {
                    String numStr = query.substring(1);
                    boolean isDigit = true;
                    for (char ch : numStr.toCharArray()) {
                        if (!Character.isDigit(ch)) {
                            isDigit = false;
                            break;
                        }
                    }
                    if (isDigit) {
                        int num = Integer.parseInt(numStr);
                        A.add(num);
                    }
                } else if (query.equals("b")) {
                    Collections.sort(A);
                } else if (query.equals("c")) {
                    Collections.reverse(A);
                } else if (query.equals("d")) {
                    output.add(String.valueOf(A.size()));
                } else if (query.equals("e")) {
                    for (int val : A) {
                        output.add(String.valueOf(val));
                    }
                } else if (query.equals("f")) {
                    A.sort(Collections.reverseOrder());
                }
            }
            System.out.println(String.join(" ", output));
        }
        sc.close();
    }
}
```

```
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem1
T == 1
Q == 4
enter 4 space-separated queries:
a10 a20 b e
10 20 111 222 333 444
PS C:\Users\Shweta\Documents\JAVACollections> 
```

Practice Problem: Ex:2

ArrayList

t 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**

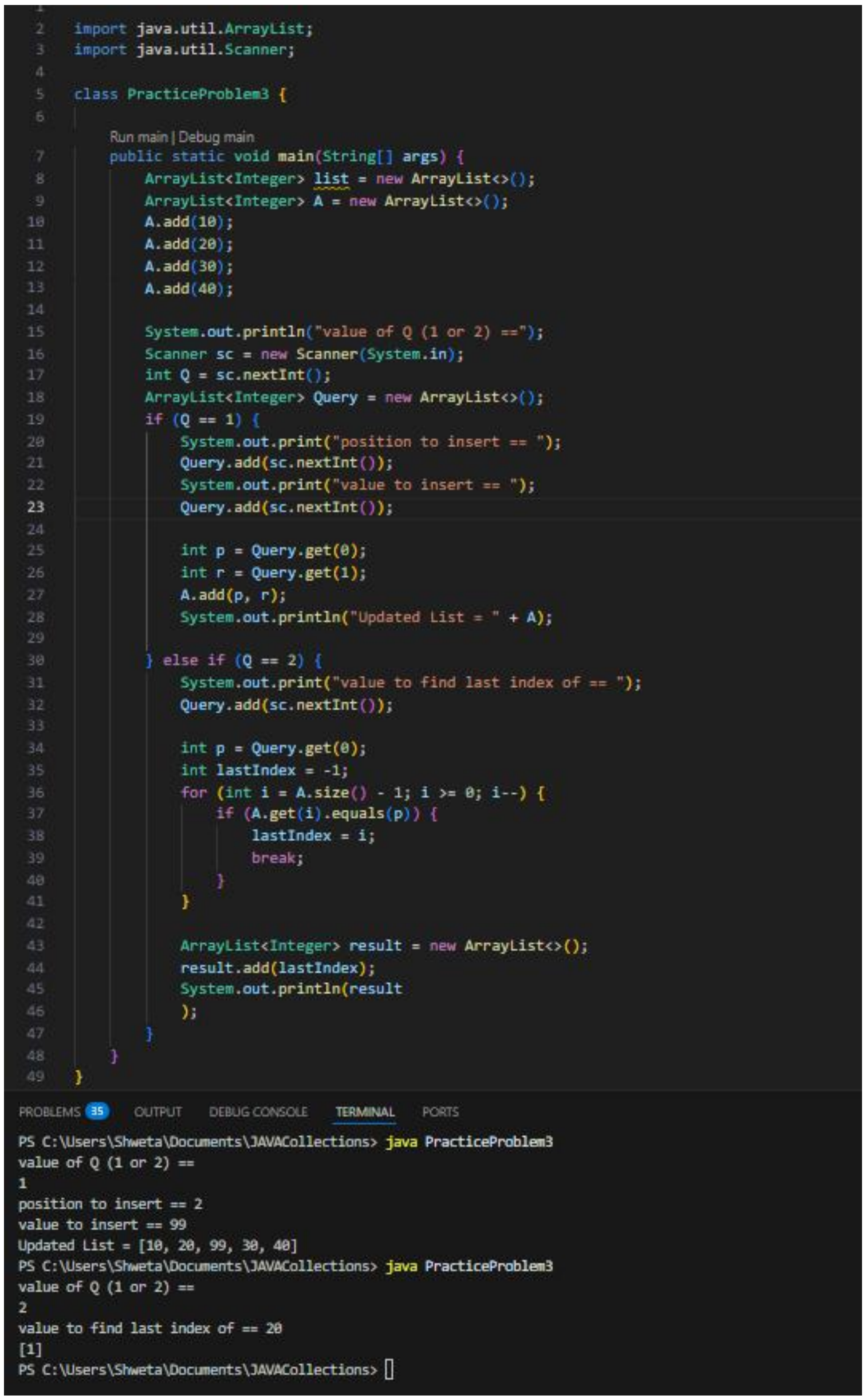
If Q = 1 then size of Query is 2,

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



```java
import java.util.ArrayList;
import java.util.Scanner;

class PracticeProblem3 {

    Run main | Debug main
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        ArrayList<Integer> A = new ArrayList<>();
        A.add(10);
        A.add(20);
        A.add(30);
        A.add(40);

        System.out.println("value of Q (1 or 2) ==");
        Scanner sc = new Scanner(System.in);
        int Q = sc.nextInt();
        ArrayList<Integer> Query = new ArrayList<>();
        if (Q == 1) {
            System.out.print("position to insert == ");
            Query.add(sc.nextInt());
            System.out.print("value to insert == ");
            Query.add(sc.nextInt());

            int p = Query.get(0);
            int r = Query.get(1);
            A.add(p, r);
            System.out.println("Updated List = " + A);

        } else if (Q == 2) {
            System.out.print("value to find last index of == ");
            Query.add(sc.nextInt());

            int p = Query.get(0);
            int lastIndex = -1;
            for (int i = A.size() - 1; i >= 0; i--) {
                if (A.get(i).equals(p)) {
                    lastIndex = i;
                    break;
                }
            }

            ArrayList<Integer> result = new ArrayList<>();
            result.add(lastIndex);
            System.out.println(result
            );
        }
    }
}
```

PROBLEMS 35 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem3
value of Q (1 or 2) ==
1
position to insert == 2
value to insert == 99
Updated List = [10, 20, 99, 30, 40]
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem3
value of Q (1 or 2) ==
2
value to find last index of == 20
[1]
PS C:\Users\Shweta\Documents\JAVACollections>
```

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.

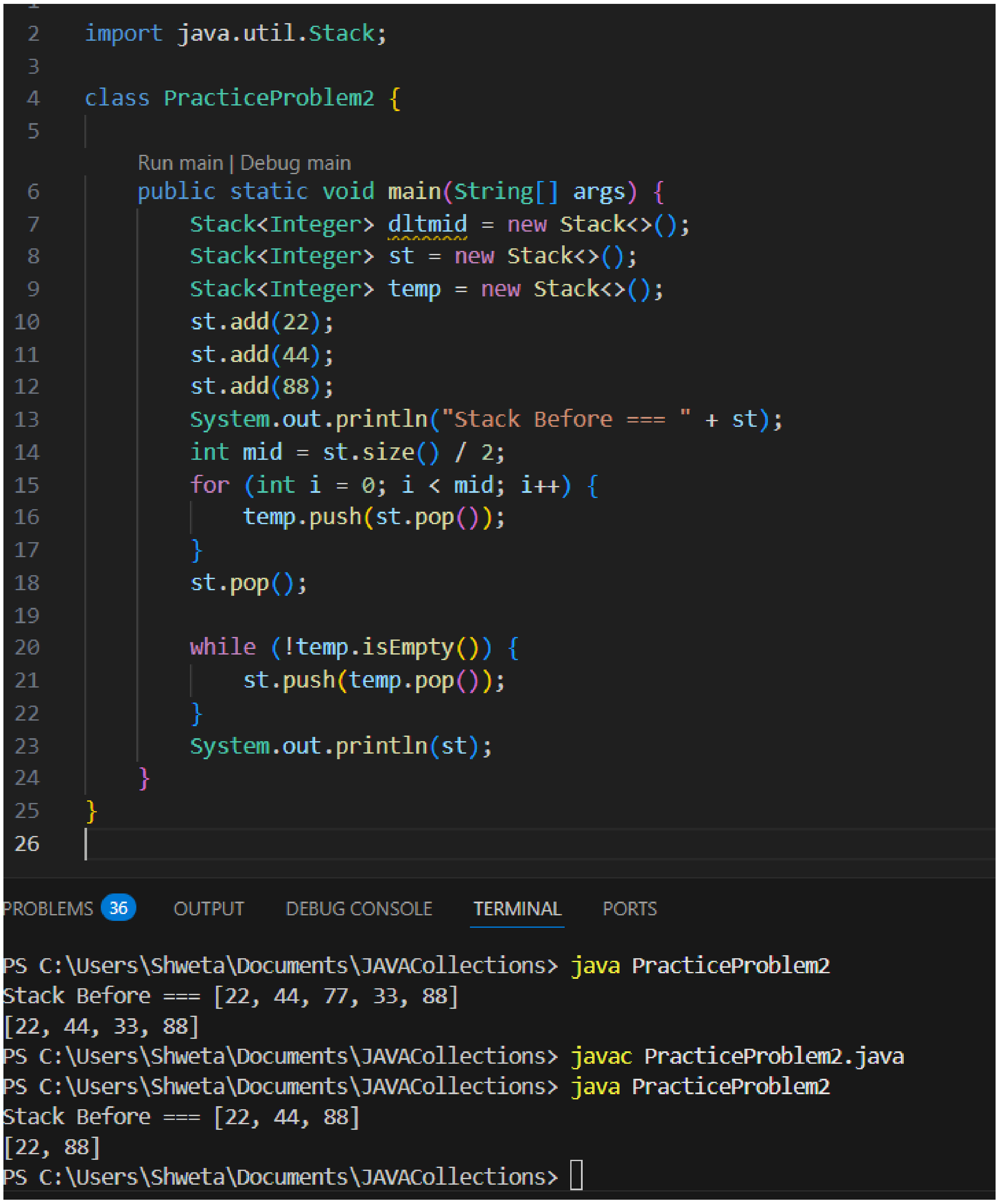
Expected Time Complexity: O(n)

Expected Auxiliary Space: O(n)

Constraints:

2 ≤ n ≤ 103

1 ≤ st[i] ≤ 104



```java
import java.util.Stack;

class PracticeProblem2 {

    Run main | Debug main
    public static void main(String[] args) {
        Stack<Integer> dltmid = new Stack<>();
        Stack<Integer> st = new Stack<>();
        Stack<Integer> temp = new Stack<>();
        st.add(22);
        st.add(44);
        st.add(88);
        System.out.println("Stack Before === " + st);
        int mid = st.size() / 2;
        for (int i = 0; i < mid; i++) {
            temp.push(st.pop());
        }
        st.pop();

        while (!temp.isEmpty()) {
            st.push(temp.pop());
        }
        System.out.println(st);
    }
}

```

```
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem2
Stack Before === [22, 44, 77, 33, 88]
[22, 44, 33, 88]
PS C:\Users\Shweta\Documents\JAVACollections> javac PracticeProblem2.java
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem2
Stack Before === [22, 44, 88]
[22, 88]
PS C:\Users\Shweta\Documents\JAVACollections> 
```

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

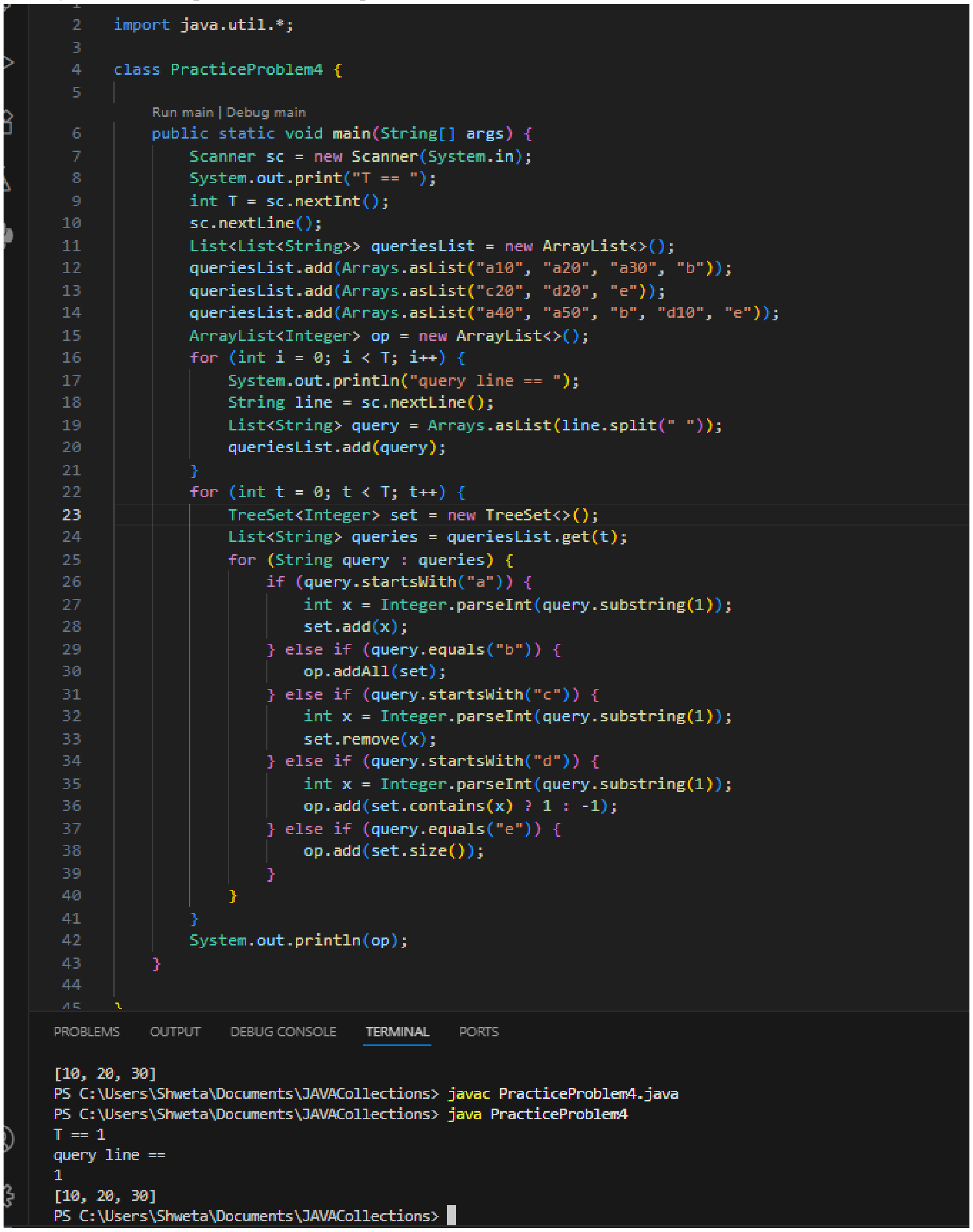
{prints the size of the set ie 2}

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.*;

class PracticeProblem4 {

    Run main | Debug main
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("T == ");
        int T = sc.nextInt();
        sc.nextLine();
        List<List<String>> queriesList = new ArrayList<>();
        queriesList.add(Arrays.asList("a10", "a20", "a30", "b"));
        queriesList.add(Arrays.asList("c20", "d20", "e"));
        queriesList.add(Arrays.asList("a40", "a50", "b", "d10", "e"));
        ArrayList<Integer> op = new ArrayList<>();
        for (int i = 0; i < T; i++) {
            System.out.println("query line == ");
            String line = sc.nextLine();
            List<String> query = Arrays.asList(line.split(" "));
            queriesList.add(query);
        }
        for (int t = 0; t < T; t++) {
            TreeSet<Integer> set = new TreeSet<>();
            List<String> queries = queriesList.get(t);
            for (String query : queries) {
                if (query.startsWith("a")) {
                    int x = Integer.parseInt(query.substring(1));
                    set.add(x);
                } else if (query.equals("b")) {
                    op.addAll(set);
                } else if (query.startsWith("c")) {
                    int x = Integer.parseInt(query.substring(1));
                    set.remove(x);
                } else if (query.startsWith("d")) {
                    int x = Integer.parseInt(query.substring(1));
                    op.add(set.contains(x) ? 1 : -1);
                } else if (query.equals("e")) {
                    op.add(set.size());
                }
            }
        }
        System.out.println(op);
    }

}
```

```
[10, 20, 30]
PS C:\Users\Shweta\Documents\JAVACollections> javac PracticeProblem4.java
PS C:\Users\Shweta\Documents\JAVACollections> java PracticeProblem4
T == 1
query line ==
1
[10, 20, 30]
PS C:\Users\Shweta\Documents\JAVACollections>
```