Part 1

Q1) A class called circle is designed as shown in the following. It contains:

• Two private instance variables: radius (of type double) and color (of type String), with a default value of 1.0 and "red", respectively.

• Two overloaded constructors.

• Two public methods: getRadius() and getArea(). Write a test program that uses Circle class and its features.

// Code

```
import java.lang.Math;

class circle {
  private double radius;
  private String color;

  circle() {
    radius = 1.0;
    color = "red";
  }

  circle(double radius, String color) {
    this.radius = radius;
    this.color = color;
  }

  double getRadius() {
    return this.radius;
  }

  double getArea() {
    return this.radius * this.radius * Math.PI;
  }
}

class makeCircle {
  public static void main(String args[]) {
    circle c1 = new circle();
    circle c2 = new circle(1.1, "Orange");

    System.out.println("For circle 1: radius: " + c1.getRadius() + " area: " +
c1.getArea());
    System.out.println("For circle 2: radius: " + c2.getRadius() + " area: " +
c2.getArea());
  }
}
```

// Output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java> javac makeCircle.java
PS C:\Users\Akshit\Documents\lumiq\WS\java> java makeCircle
For circle 1: radius: 1.0 area: 3.141592653589793
For circle 2: radius: 1.1 area: 3.8013271108436504
```

Q2) Create a class called Date that includes three pieces of information as instance variables—a month (type int), a day (type int), and a year (type int). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method display date that displays the month, day, and year separated by forward slashes (/). Write a test application named DateTest that demonstrates class dates capabilities.

//Code

```
class date {
  private int day, month, year;

  date(int day, int month, int year) {
    this.day = day;
    this.month = month;
    this.year = year;
  }

  int getDay() {
    return this.day;
  }

  int getMonth() {
    return this.month;
  }

  int getYear() {
    return this.year;
  }

  void setDay(int day) {
    this.day = day;
  }

  void setMonth(int month) {
    this.month = month;
  }

  void setYear(int year) {
    this.year = year;
  }
```

```
  void displayDate() {
    System.out.println("Date: " + this.day + "/" + this.month + "/" +
this.year);
  }
}

class dateTest {
  public static void main(String args[]) {
    try {

      // create file
      File newFile = new File("dateTest.txt");
      newFile.createNewFile();
    } catch (Exception e) {
      System.out.println("File cannot created");
    }

    // put content in file
    try {
      // make instance of writer
      FileWriter filewriter = new FileWriter("dateTest.txt");

      // write
      filewriter.write("18 2 2022\n");
      filewriter.write("1 2 2023\n");
      filewriter.write("3 3 2023");

      // close
      filewriter.close();
    } catch (Exception e) {
      System.out.println("unable to write in file");
    }

    // read file
    try {
      // get file
      File fileRead = new File("dateTest.txt");

      // make reader scanner
      Scanner lineReader = new Scanner(fileRead);

      // traverse each line
      while (lineReader.hasNext()) {

        // get date
        int day = Integer.parseInt(lineReader.next()), month =
Integer.parseInt(lineReader.next()),
            year = Integer.parseInt(lineReader.next());
        date d1 = new date(day, month, year);
```

```
        System.out.print("Initial ");
        d1.displayDate();

        d1.setDay(12);
        d1.setMonth(3);
        d1.setYear(2023);
        System.out.println("Final Date: " + d1.getDay() + "/" + d1.getMonth()
+ "/" +
            d1.getYear());
      }
      // close reader
      lineReader.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

// Output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java> cd "c:\Users\Akshit\Documents\lumiq\WS\java\" ; if ($?) { javac
makeDate.java } ; if ($?) { java makeDate }
Initial Date: 18/2/2022
Final Date: 12/3/2023
Initial Date: 1/2/2023
Final Date: 12/3/2023
Initial Date: 3/3/2023
Final Date: 12/3/2023
```

3) Write an application that inputs three integers from the user and displays the sum, average, product, smallest and largest of the numbers.

//Code

```
import java.util.Scanner;
import java.lang.Math;

class numbers {
  public static void main(String arg[]) {
    Scanner sc = new Scanner(System.in);

    int n1 = sc.nextInt(), n2 = sc.nextInt(), n3 = sc.nextInt();
    System.out.println("sum: " + (n1 + n2 + n3));
    System.out.println("avg: " + (n1 + n2 + n3) / 3);
    System.out.println("product: " + (n1 * n2 * n3));
    System.out.println("smallest: " + Math.min(Math.min(n1, n2), n3));
    System.out.println("largest: " + Math.max(Math.max(n1, n2), n3));

    sc.close();
  }
}
```

//Output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java> cd "c:\Users\Akshit\Documents\lumiq\WS\java\" ; if ($?) { javac
numbers.java } ; if ($?) { java numbers }
21 41 11
sum: 73
avg: 24
product: 9471
smallest: 11
largest: 41
```

4) Create subclass Cylinder which is derived from the Circle

- Cylinder class contains one instance variable i.e height (Type = double)
- Constructors in the cylinder class and also invoke the constructor of Circle class
- Invoke the circle class variables and methods by the instance of the Cylinder class
- Create one public method: getVolume(). write a test program to test Cylinder class and its features.

//code

```
import java.lang.Math;

class circle {
  protected double radius;
  protected String color;

  circle() {
    System.out.println("Circle constructor");
    radius = 2.0;
    color = "red";
  }

  circle(double radius, String color) {
    System.out.println("Circle constructor");
    this.radius = radius;
    this.color = color;
  }

  double getRadius() {
    return this.radius;
  }

  double getArea() {
    return this.radius * this.radius * Math.PI;
  }
}

class cylinder extends circle {
  private double height;

  cylinder() {
    System.out.println("Cylinder constructor");
```

```
    height = 3.0;
  }

  double getVolume() {
    return Math.PI * this.height * this.radius * this.radius;
  }
}

class makeCylinder {
  public static void main(String arg[]) {
    System.out.println("constructor calls: ");
    cylinder c1 = new cylinder();
    System.out.println();
    System.out
        .println("Circle class : \nVariables: Radius - " + c1.radius +
"\nFunction: getArea() - " + c1.getArea());
    System.out.println("Volume of cylinder: " + c1.getVolume());
  }
}
```

//output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java> cd "c:\Users\Akshit\Documents\lumiq\WS\java\" ; if ($?) { javac
makeCylinder.java } ; if ($?) { java makeCylinder }
constructor calls:
Circle constructor
Cylinder constructor

Circle class :
Variables: Radius - 2.0
Function: getArea() - 12.566370614359172
Volume of cylinder: 37.69911184307752
```

Part 2

Q1 Given a string s, recursively remove adjacent duplicate characters from the string. The output string should not have any adjacent duplicates.

Example: -

Input1: azxxzy

Output1: ay

Input2: caaabbbaacdddd

Output2: Empty String

//code

```
class part2_1 {
    static String func(String input, int prev, int idx) {
        // base
        if (idx == input.length() - 1) {
            if (input.charAt(idx) == input.charAt(prev))
```

```
                return input.substring(0, prev);
            else {
                if (prev > 0)
                    return input.substring(0, prev) + input.substring(idx);
                else
                    return input.substring(idx);
            }
        }

        // recur
        if (input.charAt(idx) != input.charAt(prev)) {
            if (idx - 1 != prev) {
                if (prev > 0)
                    return func(input.substring(0, prev) +
input.substring(idx), prev - 1, prev);
                else
                    return func(input.substring(idx), 0, 1);
            } else
                return func(input, prev + 1, idx + 1);
        } else
            return func(input, prev, idx + 1);
    }

    static String rmvAdjacentString(String s) {
        String ans = func(s, 0, 1);
        return ans;
    }

    public static void main(String args[]) {
        String s = "azxxzy";
        System.out.println(rmvAdjacentString(s));
    }
}
```

//output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java>  c:; cd 'c:\Users\Akshit\Documents\lumiq\WS\java'; & 'C:\Program Files\Java\jdk-18.0.1\b
de\User\workspaceStorage\4caaa5545204fbf2c4a3bc9cb19b9328\redhat.java\jdt_ws\java_72725195\bin' 'part2_1'
ay
```

Q2) Given an array of size n and an integer number k, find the count of distinct numbers from a subarray of size k. Example: -

Input1: arr[] = {1, 2, 1, 3, 4, 2, 3}; k = 4; n=7

Output1: 3,4,4,4

{1, 2, 1, 3} =>3

{2, 1, 3, 4} =>4

{1, 3, 4, 2} =>4

{3, 4, 2, 3} =>3

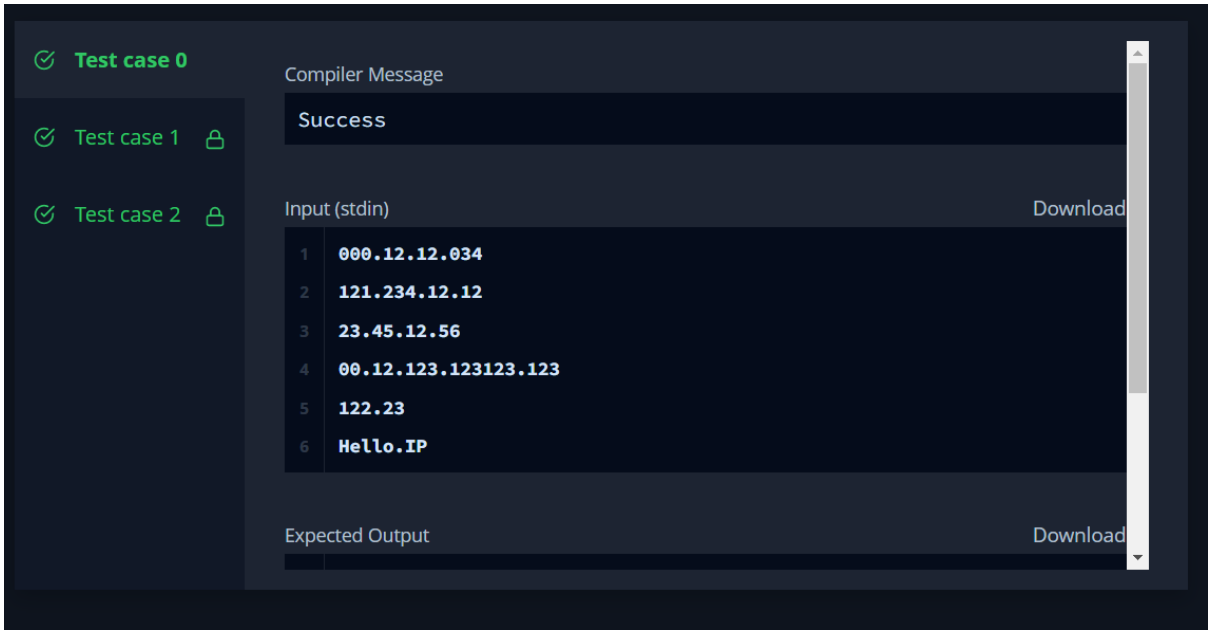
//code

```java
import java.util.HashMap;

class part2_2 {
  static int[] maxDistinctNumbInSubarrOfSizeK(int arr[], int k) {
    int res[] = new int[arr.length - k + 1];
    HashMap<Integer, Integer> hmap = new HashMap<Integer, Integer>();

    int prev = 0, idx = 0;

    for (int i : arr) {
      hmap.put(i, idx);

      if (idx - prev + 1 == k) {
        res[idx - k + 1] = hmap.size();
        if (hmap.containsKey(arr[prev]) && hmap.get(arr[prev]) <= idx - k + 1)
          hmap.remove(arr[prev]);

        prev++;
      }
      idx++;
    }

    return res;
  }

  public static void main(String args[]) {
    int k = 4;
    int arr[] = { 1, 2, 1, 3, 4, 2, 3 };
    int res[] = maxDistinctNumbInSubarrOfSizeK(arr, k);
    for (int i : res)
      System.out.println(i);
  }
}
```

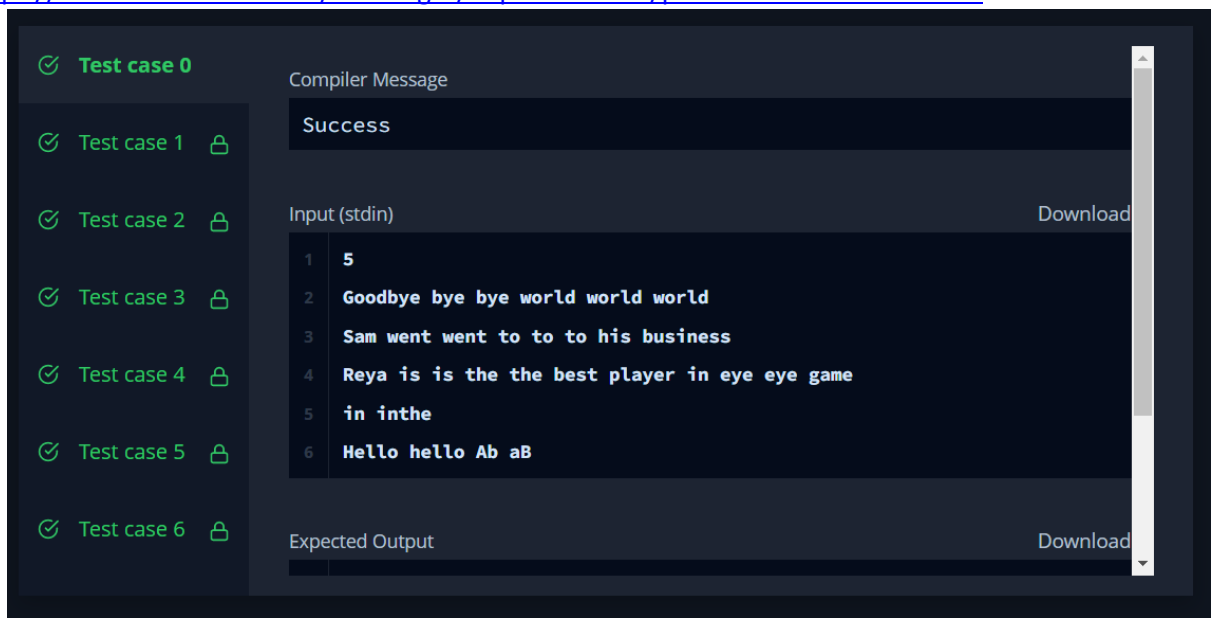
//output

```
PS C:\Users\Akshit\Documents\lumiq\WS\java> c:; cd 'c:\Users\Akshit\Documents\lumiq\WS\java'; & 'C:\Program Files\Java\jdk-18.0.1\bin\java.exe' '-XX:+ShowCodeDetailsI
ppData\Roaming\Code\User\workspaceStorage\4caaa5545204fbf2c4a3bc9cb19b9328\redhat.java\jdt_ws\java_72725195\bin' 'part2_2'
3
4
4
3
PS C:\Users\Akshit\Documents\lumiq\WS\java>
```

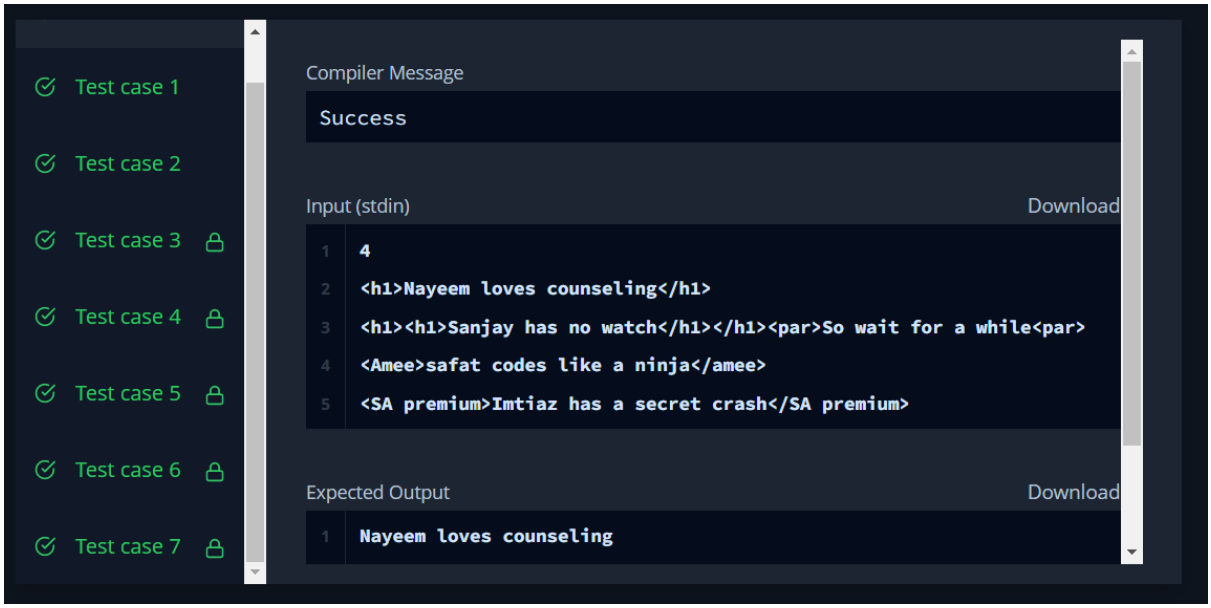
1) [Java Regex | HackerRank]



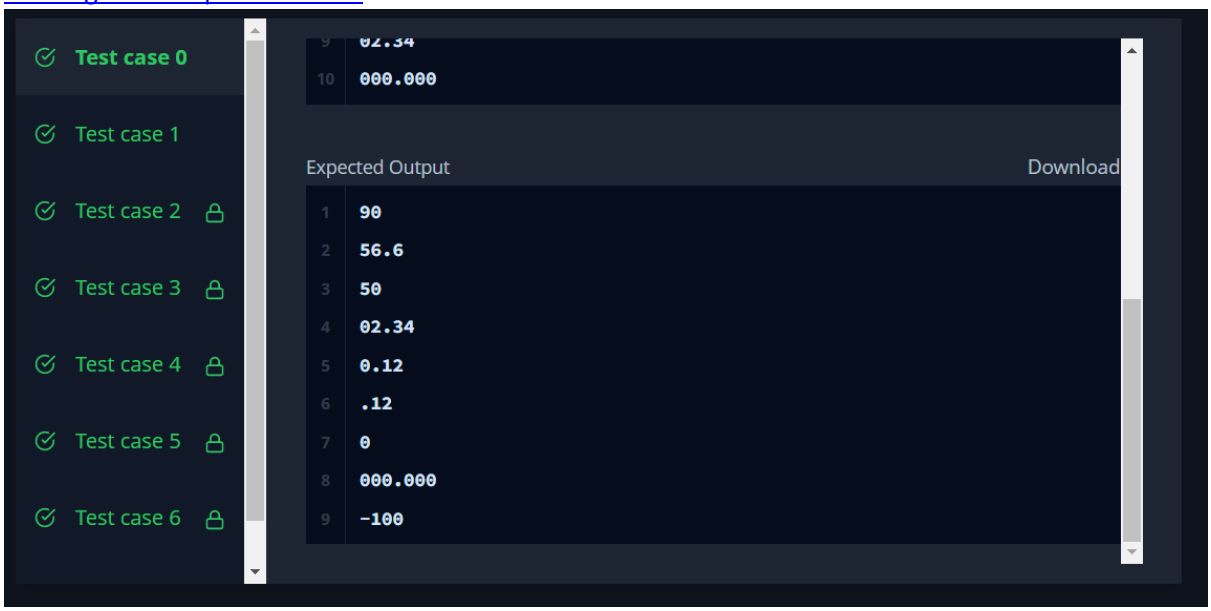
2) [https://www.hackerrank.com/challenges/duplicate-word/problem?isFullScreen=true]



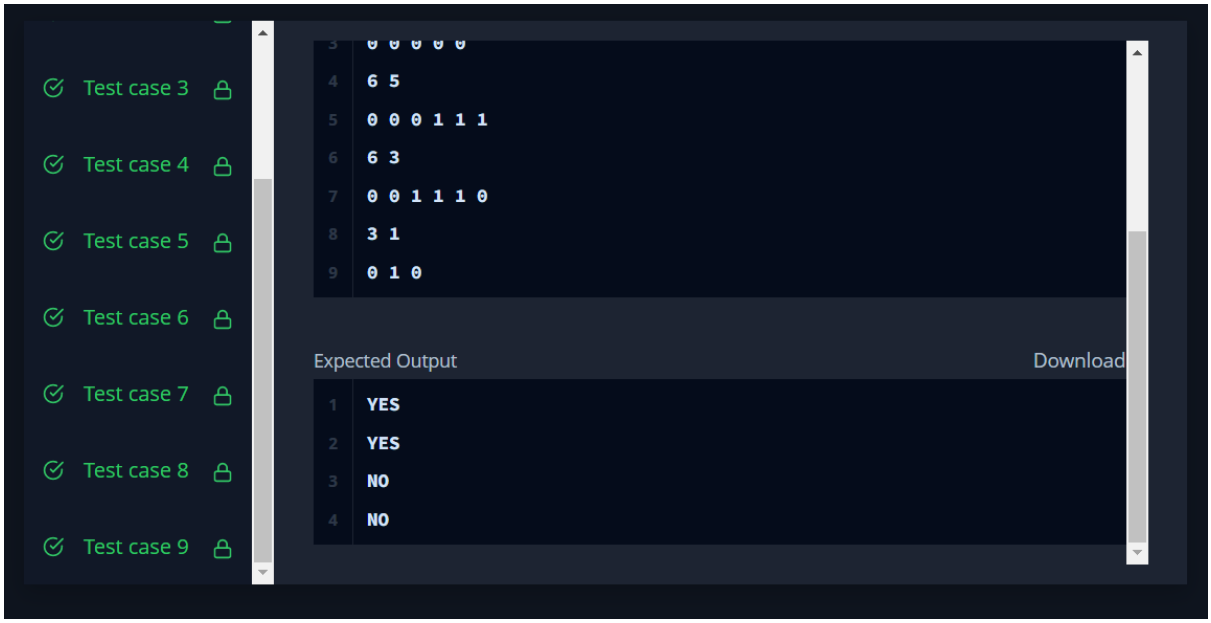
3) [Tag Content Extractor | HackerRank]



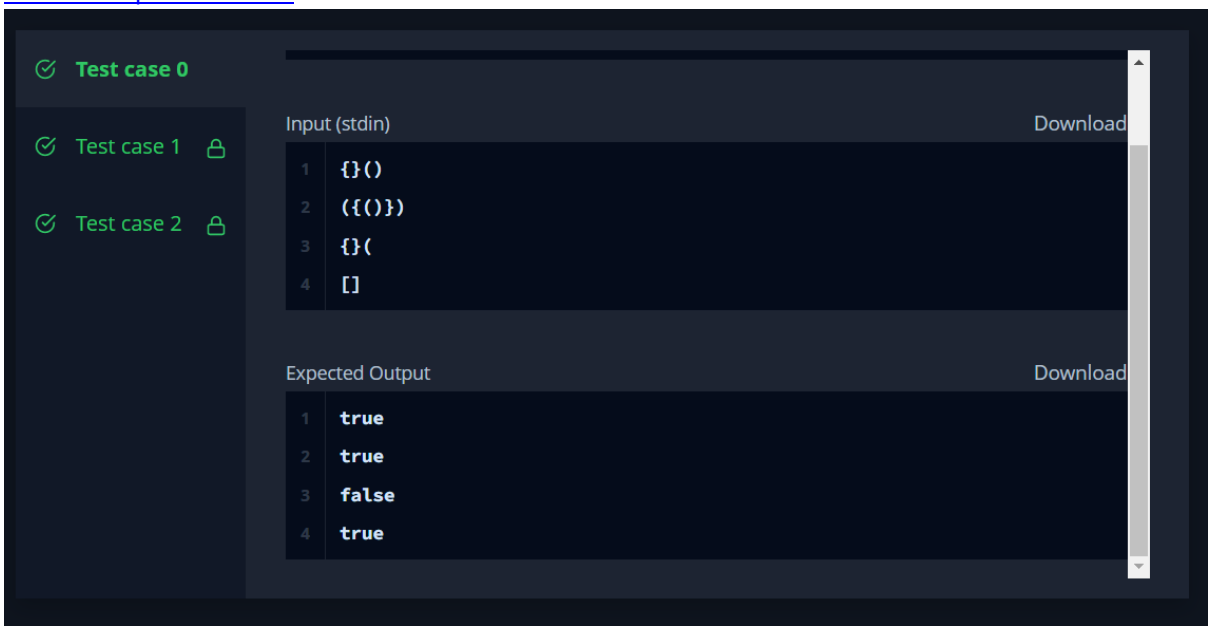
4) [Java BigDecimal | HackerRank]



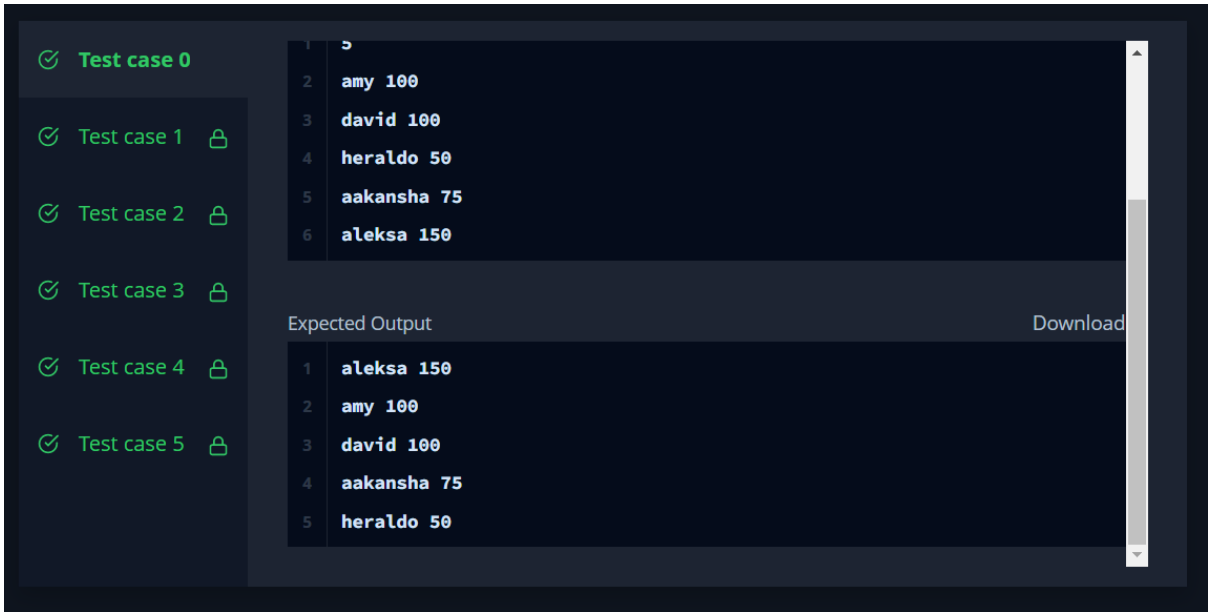
5) [Java 1D Array (Part 2) | HackerRank]



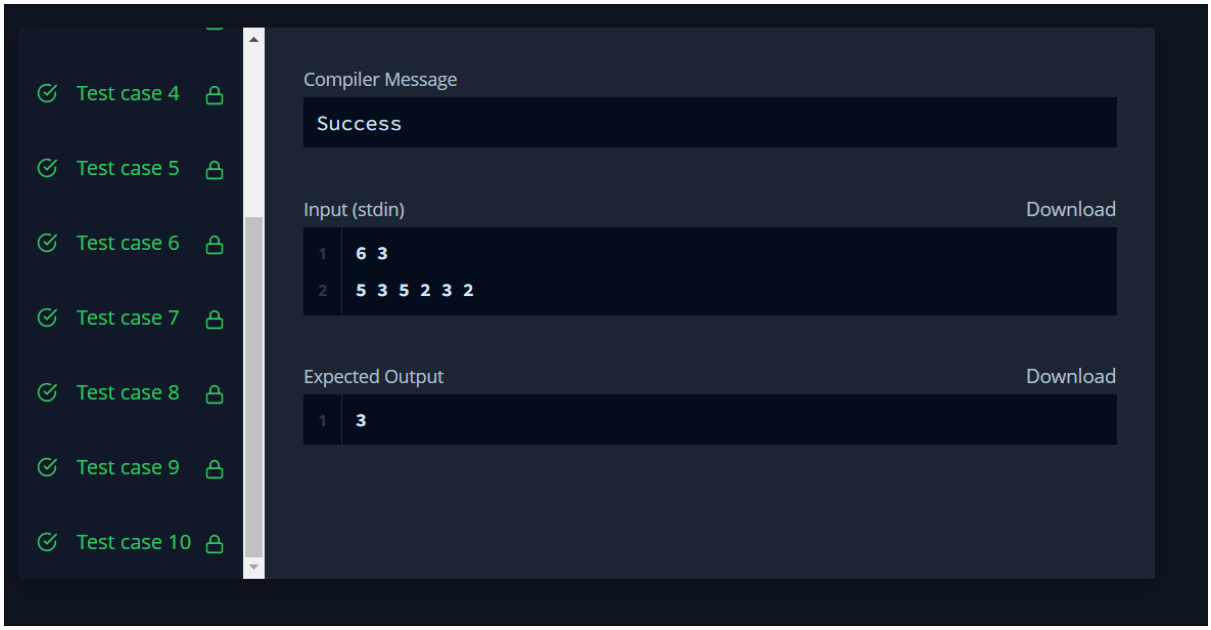
6) [Java Stack | HackerRank]



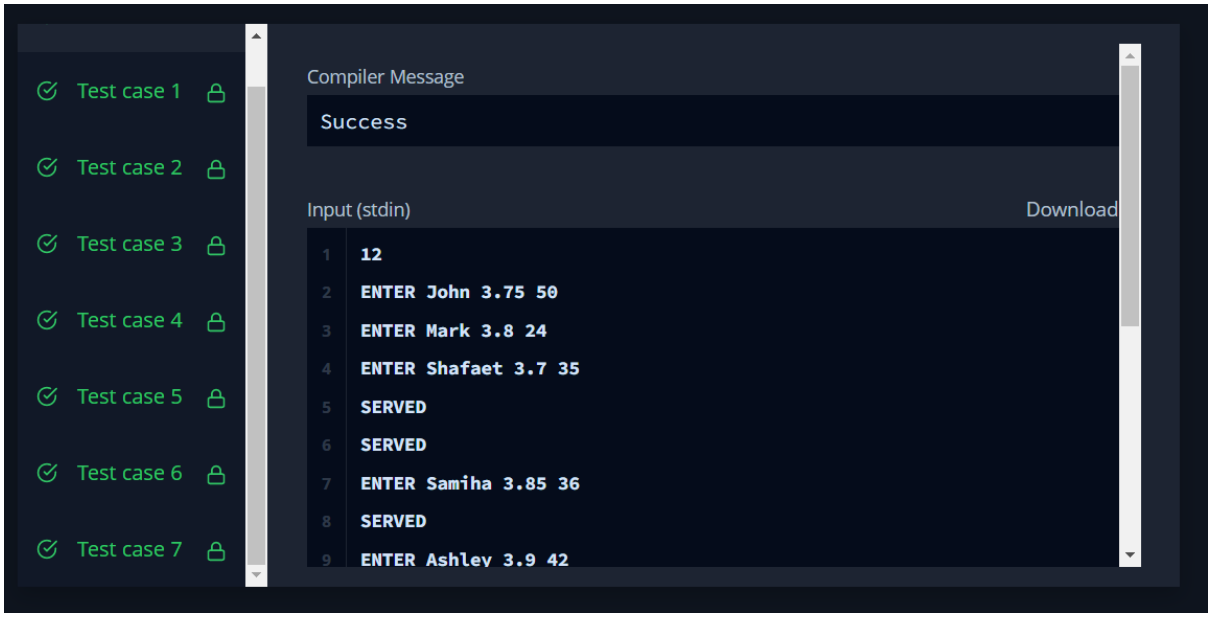
7) [Java Comparator | HackerRank]



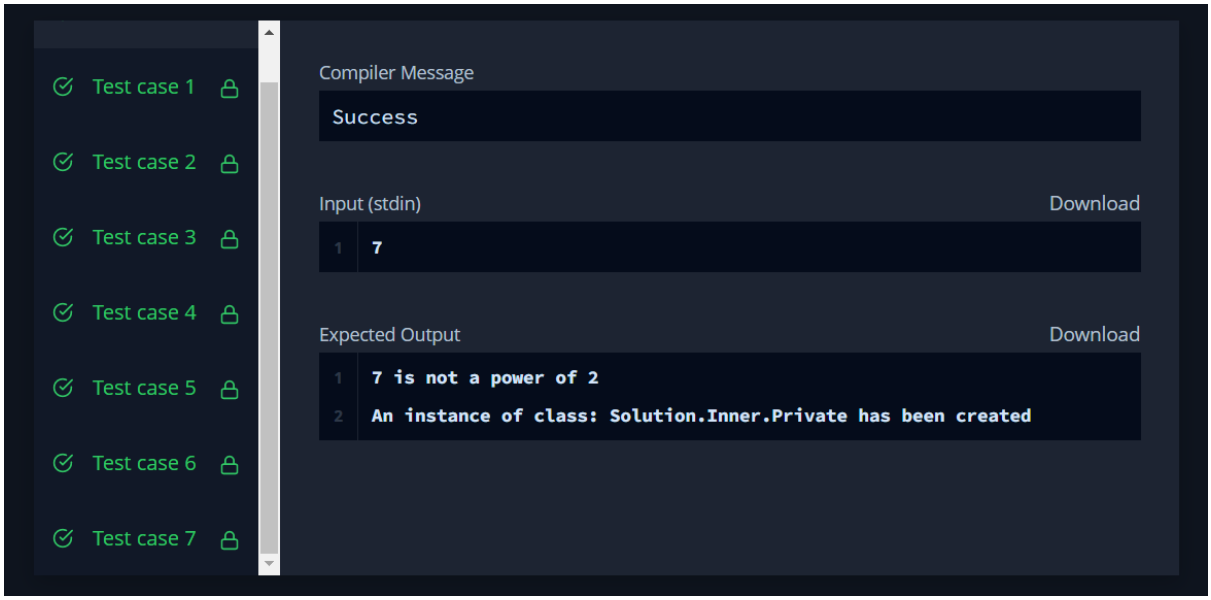
8) [Java Dequeue | HackerRank]



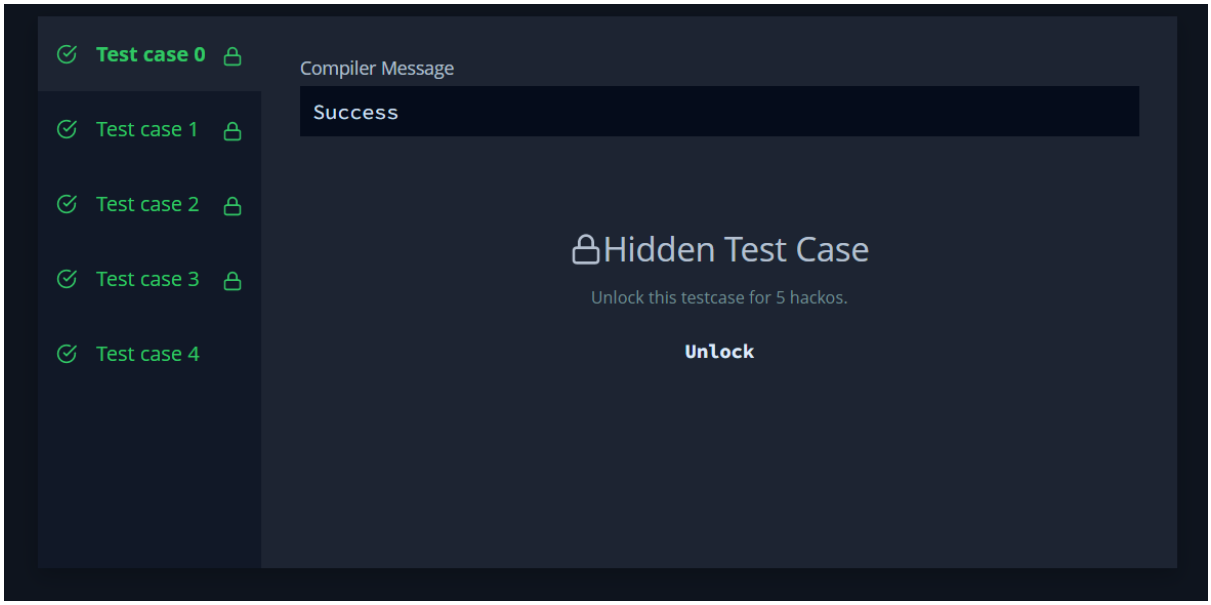
9) [Java Priority Queue | HackerRank]



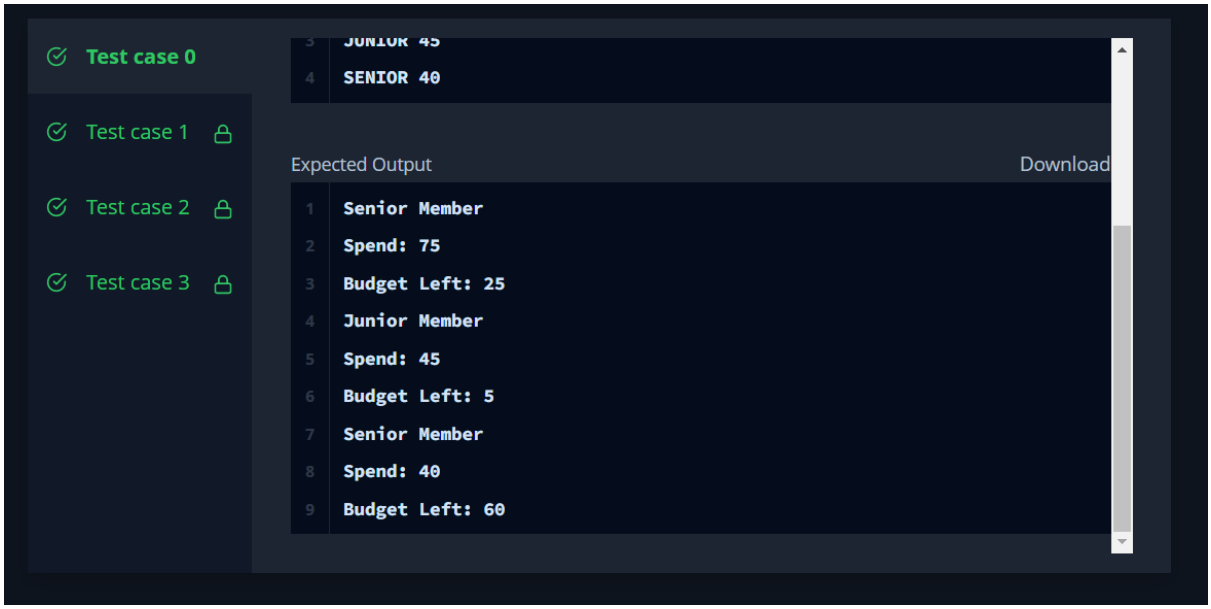
10) [Can You Access? | HackerRank]



11) [Prime Checker | HackerRank]

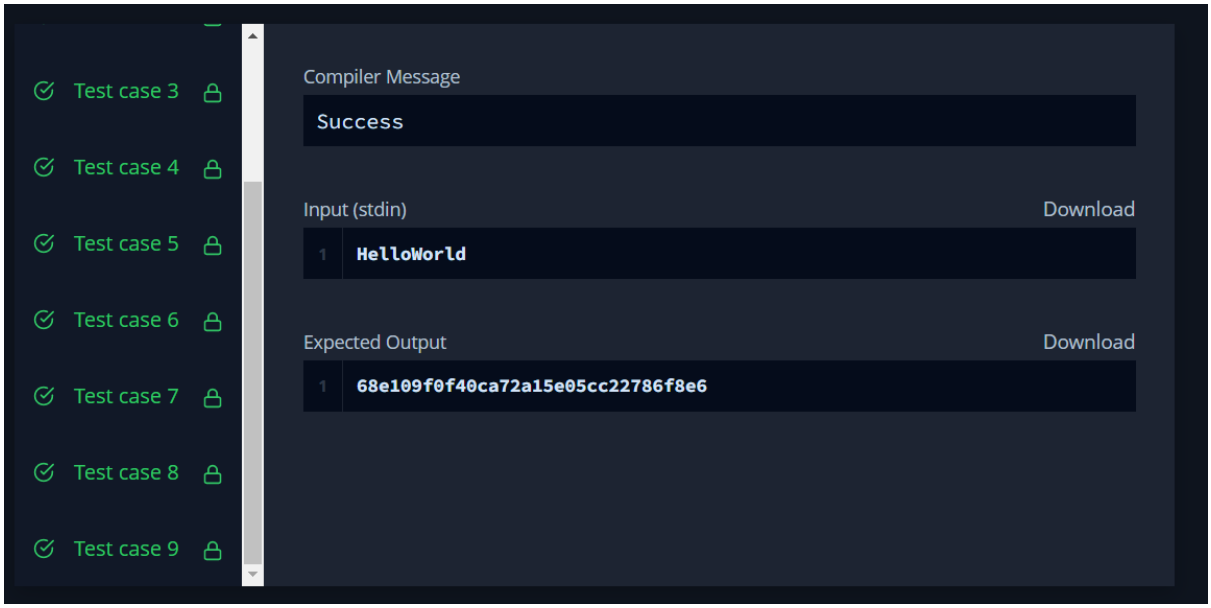


12) [Java Annotations | HackerRank]

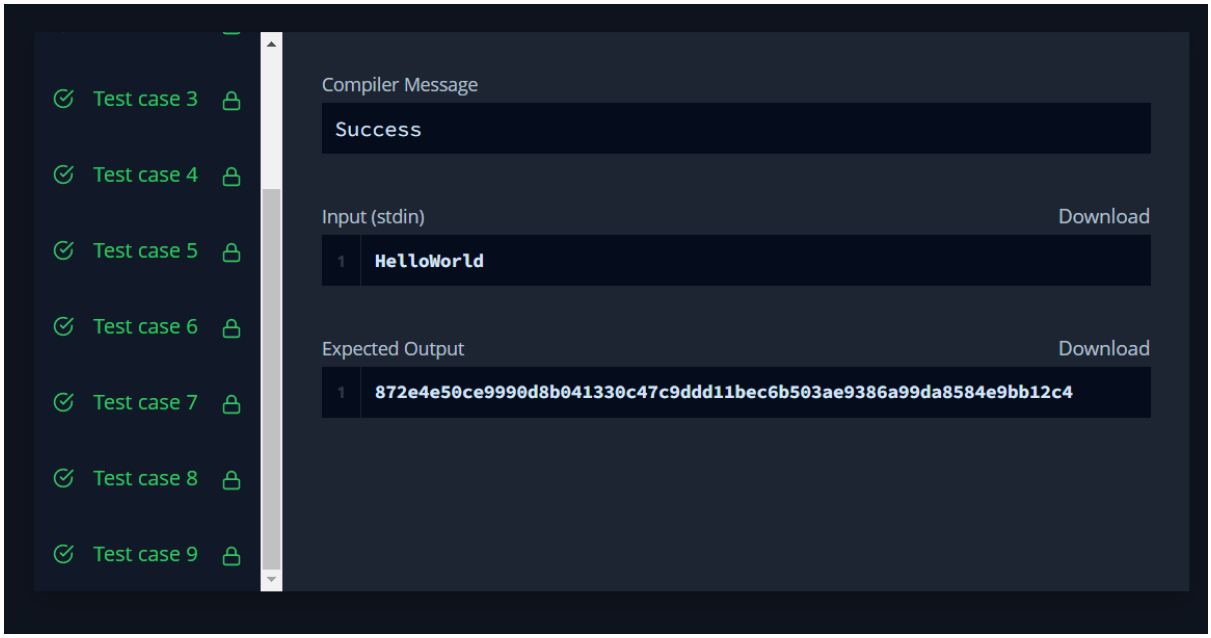


13) [Java Lambda Expressions | HackerRank]

14) [Java MD5 | HackerRank]



15) [Java SHA-256 | HackerRank]



Hacker rank solution github link

[AkshitLumiq/Java-Assignment (github.com)]