Started on	Wednesday, 3 January 2024, 2:03 PM
State	Finished
Completed on	Wednesday, 3 January 2024, 3:18 PM
Time taken	1 hour 14 mins

Question **1**

Correct

Marked out of 25.00

Write a program that returns the last digit of the given number. Last digit is being referred to the least significant digit i.e. the digit in the ones (units) place in the given number.

The last digit should be returned as a positive number.

For example,

if the given number is 197, the last digit is 7

if the given number is -197, the last digit is 7

For example:

Input	Result
197	7
-197	7

Answer: (penalty regime: 0 %)

```
import java.util.*;
public class UnitDigit
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        if(n>0)
        {
            System.out.print(n%10);
        }
        else
        {
            System.out.print(-n%10);
        }

    }
}
```

	Input	Expected	Got	
✔	197	7	7	✔
✔	-197	7	7	✔

Passed all tests! ✔

Question **2**

Incorrect

Marked out of 25.00

You are provided with a string which has a sequence of 1's and 0's.

This sequence is the encoded version of a English word. You are supposed write a program to decode the provided string and find the original word.

Each alphabet is represented by a sequence of 1s.

This is as mentioned below:

A : 1

B : 11

C : 111

D : 1111

E : 11111

F : 111111

G : 1111111

and so on upto Z having 26 1's (11111111111111111111111111).

The sequence of 1's in the encoded form are separated by a single 0 which helps to distinguish between 2 letters.

Example 1:

input1: 101101110

The decoded string (original word) will be: ABC

Example 2:

input1: 11111111011111011111111111101111111111110111111111111111110

The decoded string will be: HELLO

Note: The decoded string must always be in UPPER case.

For example:

Input	Result
101101110	ABC
11111111011111011111111111101111111111110111111111111111110	HELLO

Answer: (penalty regime: 0 %)

```
import java.util.*;
public class Decode
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        String s=sc.next();
        int c=0;
        char[] al=new char[26];


        for(int i=0;i<s.length();i++)
        {
            if(s.charAt(i)=='1')
            {
                c+=1;
            }
            else
            {
                System.out.print(c);
                c=0;
                continue;
            }
        }
    }
}
```

	Input	Expected	Got	
✖	101101110	ABC	123	✖
✖	11111111011111011111111111011111111111101111111111111110	HELLO	85121215	✖

Your code must pass all tests to earn any marks. Try again.

Show differences

Question **3**

Correct

Marked out of 25.00

Given,

the total levels (rows) in a hill pattern as input1,

the weight of the head level (first row) as input2, and

the weight increments of each subsequent row as input3.

You are expected to find the TOTAL weight of the hill pattern.

“Total levels” represents the number of rows in the pattern.

“Head level” represents the first row.

“Weight of a level” represents the value of each star (asterisk) in that row.

Note that the first row will have the weight of the head level, and the weight of each subsequent row will keep increasing by the specified “weight increment”.

The hill patterns will always be of the below format, starting with 1 star * at head level and increasing 1 star at each level till level N. From the second level (second row) a hash # also gets added to the pattern.

*

#

##*

##*#*

##*#*#*

. . . and so on till level N.

While the weight of a start * is equal to the weight of the current level (current row), the weight of the hash # is equal to the weight of the previous level (previous row).

Let us see a couple of examples:

Example 1:

Given,

the total levels (total rows) in a hill pattern = 5 (input1)

the weight of the head level (first row) = 10 (input2)

the weight increments of each subsequent level = 2 (input3)

Then, the total weight of the hill pattern will be calculated as = 10 + (12 + 10 + 12) + (14 + 12 + 14 + 12 + 14) + (16 + 14 + 16 + 14 + 16 + 14 + 16) + (18 + 16 + 18 + 16 + 18 + 16 + 18 + 16 + 18) = 10 + 34 + 66 + 106 + 154 = 370

Example 2:

Given,

the total levels (total rows) in a hill pattern = 4 (input1)

the weight of the head level (first row) = 1 (input2)

the weight increments of each subsequent level = 5 (input3)

Then, the total weight of the hill pattern will be = 1 + (6 + 1 + 6) + (11 + 6 + 11 + 6 + 11) + (16 + 11 + 16 + 11 + 16 + 11 + 16) = 1 + 13 + 45 + 97 = 156

Observe the weight of star *: Please observe that the weight of star * in first row is 10, in second row it increases by 2 and becomes 12, in third row it increases by 2 and becomes 14, in fourth row it increases by 2 and becomes 16 and so on . . .

Observe the weight of hash #: Please observe that the weight of hash # in each row is equal to the weight of the star * in the previous row.

For example:

Input	Result
5 10 2	370

Input	Result
4 1 5	156

Answer: (penalty regime: 0 %)

```
import java.util.*;
public class Pattern{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int t=sc.nextInt();
        int w=sc.nextInt();
        int inc=sc.nextInt();
        int tw=0,h=w;
        for(int i=1;i<=t;i++)
        {
            if(i==1)
            {
                tw+=w;
            }
            else
            {
                tw+=(w+inc)*i+(h*(i-1));
                w=w+inc;
                h=h+inc;
                //System.out.print(h+" ");
            }

        }
        System.out.print(tw);
    }
}
```

	Input	Expected	Got	
✔	5 10 2	370	370	✔
✔	4 1 5	156	156	✔

Passed all tests! ✔

Question **4**

Incorrect

Marked out of 25.00

Mamta has received an array of positive numbers. The numbers in the array are in the range from 1 to 250. She has to generate a password using the below mechanism:

The password should consist of THREE parts.

Password = (Part 1)(Part 2)(Part3)

Part 1 = (Number that occurs HIGHEST number of times in the array)

Part 2 = (Number that occurs SECOND HIGHEST number of times in the array)

Part 3 = (Number that occurs LEAST number of times in the array)

Note regarding Part 1 – If more than one number occurs HIGHEST number of times, we must choose the LARGEST of them.

Note regarding Part 2 – If more than one number occurs SECOND HIGHEST number of times, we must choose the LARGEST of them.

Note regarding Part 3 – If more than one number occurs LEAST number of times, we must choose the SMALLEST of them.

Example 1:

If input1 = 10 (representing the number of elements in the array) and the array input2 = {12, 2, 36, 10, 217, 36, 5, 36, 15, 10}.

Then, the password should be formed as follows by combining the 2 parts:

Password = (36)(10)(2) = 36102

Explanation:

In the given array input1,

36 occurs 3 times (So 36 is the Number that occurs HIGHEST number of times in the array).

10 occurs 2 times (So 10 is the Number that occurs SECOND HIGHEST number of times in the array).

There are five numbers (217, 15, 12, 5 and 2) that occur 1 time each and 2 is the lowest of these (so 2 is the number that occurs LEAST number of times in the array).

Therefore, Password = (36)(10)(2) = 36102

Example 2:

input1 = 16 (representing that the array contains 16 numbers)

input2 = {5, 123, 12, 45, 62, 77, 89, 23, 12, 14, 11, 14, 12, 90, 89, 12}

Then, Password will be = (12)(89)(5) = 1289

For example:

Input	Result
10 12 2 36 10 217 36 5 36 15 10	36102
16 5 123 12 45 62 77 89 23 12 14 11 14 12 90 89 12	1289

Answer: (penalty regime: 0 %)

```
import java.util.*;
public class Password
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<Integer> a=new ArrayList<Integer>();
        ArrayList<Integer> f=new ArrayList<Integer>();

        for(int i=0;i<n;i++)
        {
            a.add(sc.nextInt());
        }
        for(int i=0;i<n;i++)
        {
            f.add(Collections.frequency(a,a.get(i)));
        }
```

```
        }

        // HashMap<Integer,Integer> t=new HashMap<Integer,Integer>();
        // for(int i=0;i<n;i++)
        // {
        //     t.put(i,Collections.frequency(a,a.get(i)));
        // }
        String pass=new String();
        int max1=0,ind=0;
        //System.out.print(f);
        //System.out.println();
        for(int i=0;i<n;i++)
        {
            if(f.get(i)>max1)
            {
                max1=f.get(i);
                ind=i;
            }
        }
        pass=pass+a.get(ind);
        int max2=0;
        for(int i=0;i<n;i++)
        {
            if(f.get(i)>max2 && f.get(i)<max1)
            {
                max2=f.get(i);
                ind=i;
            }
        }
        pass=pass+a.get(ind);
        int min1=max1;
        ArrayList<Integer> l=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            if(f.get(i)<min1)
```

	Input	Expected	Got	
✔	10 12 2 36 10 217 36 5 36 15 10	36102	36102	✔
✖	16 5 123 12 45 62 77 89 23 12 14 11 14 12 90 89 12	1289	12895	✖

Your code must pass all tests to earn any marks. Try again.

Show differences