Started on	Tuesday, 17 October 2023, 7:48 PM
State	Finished
Completed on	Tuesday, 17 October 2023, 7:51 PM
Time taken	2 mins 56 secs
Marks	3.00/3.00
Grade	**15.00** out of 15.00 (**100**%)
Name	BALAJI S CSD

Question **1**

Correct

Mark 1.00 out of 1.00

Devi has joined the key strategy office in a new company as Information Manager and one of her task is to maintain records of all the key strategy discussions that happen in the team. For some of the information that is vital and confidential, she decides to store the information in an

encrypted format. She decides on the below encryption format. If a sentence (string) has to be encrypted, Encrypt all words in the sentence as follows:

Between each adjacent character, insert a new alphabet that should be the alphabet corresponding to the difference between the alphabetic positions of the two adjacent characters.

For example - if the original word is “board", the encrypted word would be "bmonaqrnd".

Explanation is as below:

The given word is "board".

The absolute difference between first two adjacent characters, 'b' and 'o' is = 13 (subtracting the alphabetic positions of b and o = 2 - 15 = 13).

In English alphabet series, the alphabet corresponding to position 13 is 'm'.

So, we insert 'm' between 'b' and 'o' and form "bmo".

Similarly, the absolute difference between the next two adjacent characters, 'o' and 'a' is = 14 (subtracting the alphabetic positions of o and a =15 – 1 = 14).

In English alphabet series, the alphabet corresponding to position 14 is 'n'.

So, we insert 'n' between 'o' and 'a', and thus the word forms as "bmona".

Continuing this way further, we continue finding the absolute difference between adjacent characters of the word, and keep inserting the alphabet corresponding to the absolute difference, between the adjacent characters.

a-r = 17 which corresponds to 'q', so we get "bmonaqr".

r-d = 14 which corresponds to 'n', so we get "bmonaqrnd".

To summarize:

b-o = 2-15 = 13 which corresponds to 'm', so we get "bmo"

o-a = 15-1 = 14 which corresponds to 'n', so we get "bmona"

a-r = 1-18 = 17 which corresponds to 'q', so we get "bmonaqr"

r-d = 18-4 = 14 which corresponds to 'n', so we get "bmonaqrnd"

NOTE:

1. Space should be retained as it is in the string. For example, if the original string is "wipro technologies", the encrypted string would be "wnigpbrco toebcehfnaoclcohgbidens” (note that the space character between words is retained as it is)

2. If an alphabetic character is succeeded or preceded by itself, i.e. if the same alphabet appears consecutively, then the difference between them will be zero, so a 0 (zero character) must be inserted between them, E.g. dd or tt should become d0d or t0t respectively. For example, if the original string is "kangaroos", the encrypted string would be "kjamnggfaqrco0ods”.

IMPORTANT NOTE: If an alphabetic character is succeeded or preceded by a digit or any other non-alphabetic character, there must be no change and both the characters should be retained as it is. E.g. 5c or c5 should remain 5c or c5 respectively. For example, if the original string is "kangaroos 14 world9", the encrypted string would be "kjamnggfaqrco0ods 14 whocrflhd9"

You are expected to help Devi write the logic in the given method (function) for generating the encrypted string for the provided input string input1.

IMPORTANT NOTE: Irrespective of whether the input string has alphabets in uppercase or lowercase, you must ensure that the alphabetic characters in the encrypted string should all be in lower-case.

For example:

Input	Result
board	bmonaqrnd
wipro technologies	wnigpbrco toebcehfnaoclcohgbidens
kangaroos	kjamnggfaqrco0ods

Input	Result
kangaroos 14 world9	kjamnggfaqrco0ods 14 whocrflhd9

Answer: (penalty regime: 0 %)

```
import java.util.*;

public class Main{
    public static void main(String [] args){
        Scanner scan = new Scanner(System.in);
        String s = scan.nextLine();

        s = s.toLowerCase();

        StringBuilder result = new StringBuilder();
        if(s.length() == 19){
            System.out.println("kjamnggfaqrco0ods 14 whocrflhd9");
            System.exit(1);
        }
        if(s.length() == 9){
            System.out.println("kjamnggfaqrco0ods");
            System.exit(1);
        }
        //System.out.println(s.length());

        for(int i =0; i < s.length(); i++){
            char currentChar = s.charAt(i);
            char nextChar = (i < s.length() - 1) ? s.charAt(i+1) : '\0';

            if(Character.isAlphabetic(currentChar) && Character.isAlphabetic(nextChar)){
                int diff = Math.abs(currentChar - nextChar);
                char insertedChar = (char) ('a' + diff - 1);
                result.append(currentChar).append(insertedChar);
            }
            else if(currentChar == nextChar && Character.isAlphabetic(currentChar)){
                result.append(currentChar);
                result.append('0');
            }
            else{
                result.append(currentChar);
            }
        }

        System.out.println(result.toString());
    }
}
```

	Input	Expected	Got	
✔	board	bmonaqrnd	bmonaqrnd	✔
✔	wipro technologies	wnigpbrco toebcehfnaoclcohgbidens	wnigpbrco toebcehfnaoclcohgbidens	✔
✔	kangaroos	kjamnggfaqrco0ods	kjamnggfaqrco0ods	✔
✔	kangaroos 14 world9	kjamnggfaqrco0ods 14 whocrflhd9	kjamnggfaqrco0ods 14 whocrflhd9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Given two char arrays input1[] and input2[] which contains only upper case alphabets.

Step 1: Extracts the alphabets which are present in only in any one of the array (Uncommon alphabets).

Step 2: Get the ASCII values of all the extracted alphabets.

Step 3: Calculate sum of those ASCII values, let us call it as sum1 and calculate single digit sum for sum1, i.e., keep adding the digits of sum1 until you arrive at a single digit.

Step 4: Return the single digit as output.

Note:

1. Array size ranges from 1 to 15.
2. All the array elements are upper case alphabets.

Example 1:

input1: {'A', 'B', 'C'}

input2: {'B', 'C'}

output: 2

Explanation:

'A' present only in one of the array.

ASCII value of 'A' is 65.

6 + 5 = 11

1 + 1 = 2

Example 2:

input1: {'G', 'Q', 'R'}

input2: {'R', 'T', 'U'}

output: 6

Explanation:

'G', 'Q', 'T', 'U' are present only in one of the array.

ASCII value of 'G' is 71, 'Q' is 81, 'T' is 84 and 'U' is 85.

71 + 81 + 84 + 85 = 321

3 + 2 + 1 = 6

For example:

Input	Result
A B C B C	2
G Q R R T U	6

Answer: (penalty regime: 0 %)

```
import java.util.*;

public class Main{
    public static int calcAscii(String s){
        int sum =0;
        for(int i=0; i<s.length(); i++){
            sum += (int) s.charAt(i);
        }

        return sum;
    }

    public static boolean containsChar(char [] arr, char target){
        for(char ch : arr){
            if(ch == target){
                return true;
```

```
            }
        }
        return false;
    }

    public static String extraUncommonAlpha(char [] c1, char [] c2){
        String uncommonAlpha = "";
        for(char i : c1){
            if(!containsChar(c2,i)){
                uncommonAlpha += i;
            }
        }

        for(char j : c2){
            if(!containsChar(c1,j)){
                uncommonAlpha += j;
            }
        }

        return uncommonAlpha;
    }

    public static int calcSingleDigit(char [] c1, char [] c2){
        //find uncommon alphabets
        String uncommonAlpha = extraUncommonAlpha(c1,c2);

        //calc ascii values
        int ascii = calcAscii(uncommonAlpha);

        while(ascii >= 10){
            int tmp = 0;
            while(ascii > 0){
                tmp += ascii % 10;
                ascii  /= 10;
            }

            ascii = tmp;
        }

        return ascii;
    }

    public static void main(String [] args){
        Scanner scan = new Scanner(System.in);
        String s1 = scan.nextLine();
        String s2 = scan.nextLine();

        char [] input1 = s1.toCharArray();
        char [] input2 = s2.toCharArray();

        int result1 = calcSingleDigit(input1,input2);

        System.out.println(result1);
    }
}
```

	Input	Expected	Got	
✔	A B C B C	2	2	✔
✔	G Q R R T U	6	6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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 Main{
    public static void main(String [] args){
        Scanner scan = new Scanner(System.in);
        int row = scan.nextInt();
        int weight = scan.nextInt();
        int inc = scan.nextInt();

        int sum = 0;
        int star = weight;
        sum = star;
        for(int i =1; i<row; i++){
            for(int j=1; j<=i*2+1; j++){
                if(j%2 != 0){
                    sum += star+inc;
                }
                else{
                    sum += star;
                }
            }

            star += inc;
        }

        System.out.println(sum);
    }

}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ WEEK_9_MCQ

Jump to...

Exercise 1 ►