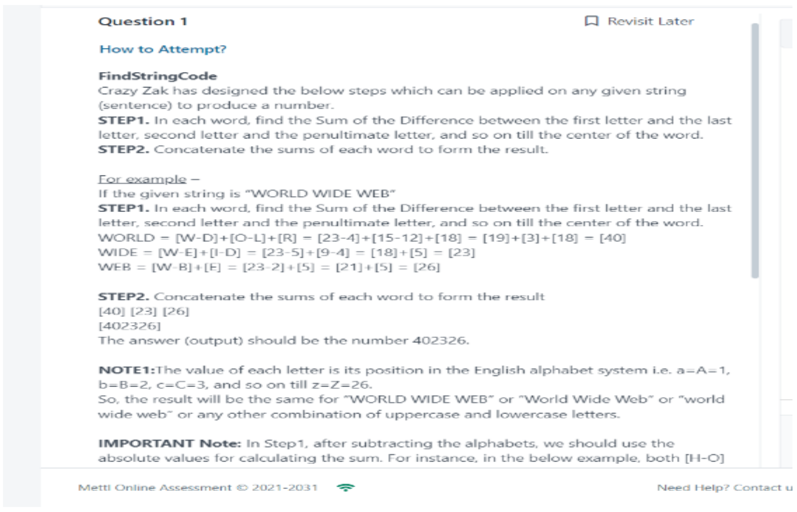


Question 1 Revisit Later

How to Attempt?

FindStringCode
Crazy Zak has designed the below steps which can be applied on any given string (sentence) to produce a number.
STEP1. In each word, find the Sum of the Difference between the first letter and the last letter, second letter and the penultimate letter, and so on till the center of the word.
STEP2. Concatenate the sums of each word to form the result.

For example –
If the given string is "WORLD WIDE WEB"
STEP1. In each word, find the Sum of the Difference between the first letter and the last letter, second letter and the penultimate letter, and so on till the center of the word.
WORLD = [W-D]+[O-L]+[R] = [23-4]+[15-12]+[18] = [19]+[3]+[18] = [40]
WIDE = [W-E]+[I-D] = [23-5]+[9-4] = [18]+[5] = [23]
WEB = [W-B]+[E] = [23-2]+[5] = [21]+[5] = [26]

STEP2. Concatenate the sums of each word to form the result
[40] [23] [26]
[402326]
The answer (output) should be the number 402326.

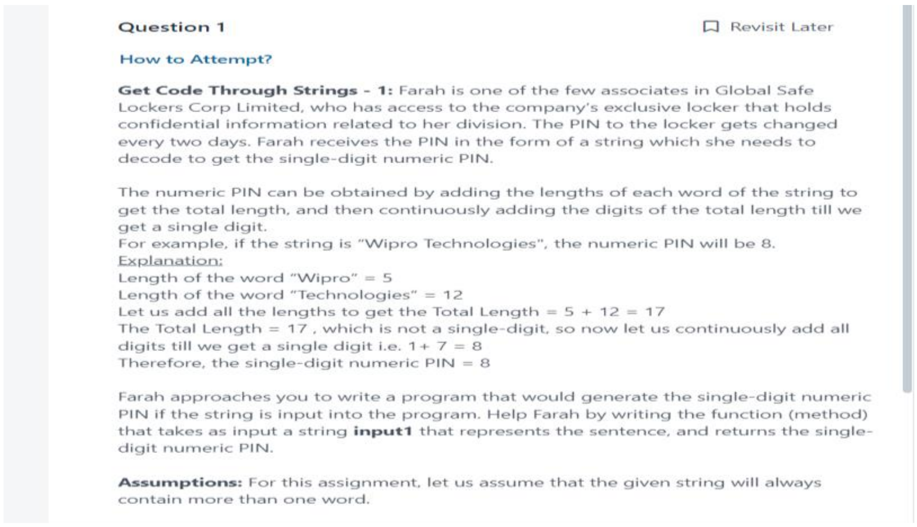
NOTE1: The value of each letter is its position in the English alphabet system i.e. a=A=1, b=B=2, c=C=3, and so on till z=Z=26.
So, the result will be the same for "WORLD WIDE WEB" or "World Wide Web" or "world wide web" or any other combination of uppercase and lowercase letters.

IMPORTANT Note: In Step1, after subtracting the alphabets, we should use the absolute values for calculating the sum. For instance, in the below example, both [H-O]

Mettl Online Assessment © 2021-2031 Need Help? Contact u

CODE 1:-

```
import java.io.*;
import java.util.*;
class UserMainCode{
public int findStringCode(String input1){
String str=input1.toUpperCase();
String word[]=str.split(" ");
String value2="";
for(int i=0;i<word.length;i++)    {
int sum=0;
for(int j=0;j<word[i].length()/2;j++)     {
int first=word[i].charAt(j);
int last=word[i].charAt(word[i].length()-1-j);
sum+=Math.abs(first-last);
}
 if(word[i].length()%2!=0)
sum+=(word[i].charAt(word[i].length()/2)-64);
String value=Integer.toString(sum);
  value2+=value;
 }
 return Integer.parseInt(value2);
 }}
```



Question 1 Revisit Later

How to Attempt?

Get Code Through Strings - 1: Farah is one of the few associates in Global Safe Lockers Corp Limited, who has access to the company's exclusive locker that holds confidential information related to her division. The PIN to the locker gets changed every two days. Farah receives the PIN in the form of a string which she needs to decode to get the single-digit numeric PIN.

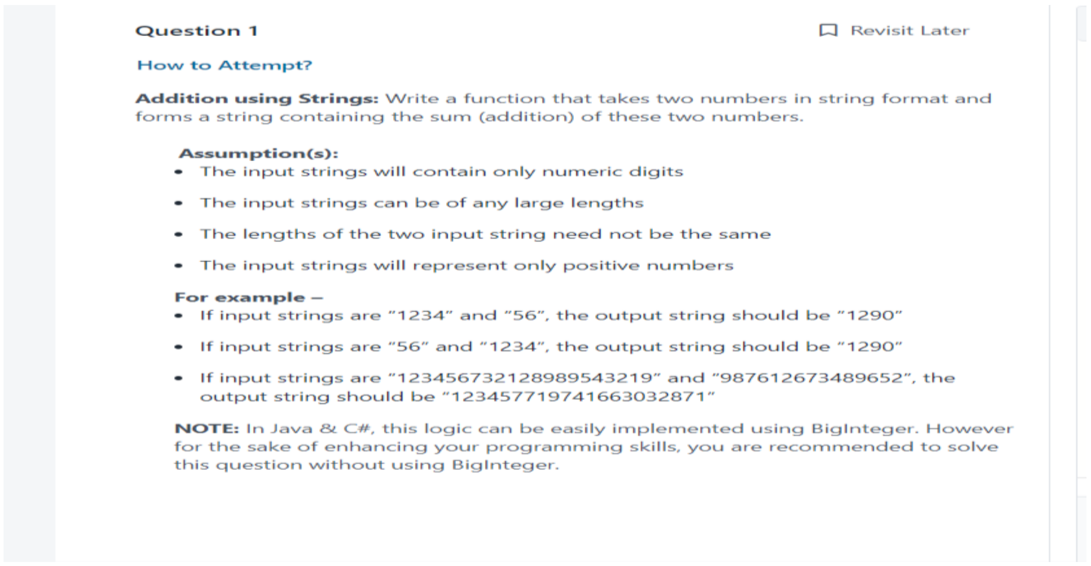
The numeric PIN can be obtained by adding the lengths of each word of the string to get the total length, and then continuously adding the digits of the total length till we get a single digit.
For example, if the string is "Wipro Technologies", the numeric PIN will be 8.
Explanation:
Length of the word "Wipro" = 5
Length of the word "Technologies" = 12
Let us add all the lengths to get the Total Length = 5 + 12 = 17
The Total Length = 17 , which is not a single-digit, so now let us continuously add all digits till we get a single digit i.e. 1+ 7 = 8
Therefore, the single-digit numeric PIN = 8

Farah approaches you to write a program that would generate the single-digit numeric PIN if the string is input into the program. Help Farah by writing the function (method) that takes as input a string **input1** that represents the sentence, and returns the single-digit numeric PIN.

Assumptions: For this assignment, let us assume that the given string will always contain more than one word.

Code2:

```
import java.io.*;
import java.util.*;
class UserMainCode{
public int getCodeThroughString(String input1){
String word[]=input1.split(" ");
int sum=0;
for(int i=0;i<word.length;i++)
{
sum+=word[i].length();
}
return (1 + (sum-1) %9);
}}
```



Question 1 Revisit Later

How to Attempt?

Addition using Strings: Write a function that takes two numbers in string format and forms a string containing the sum (addition) of these two numbers.

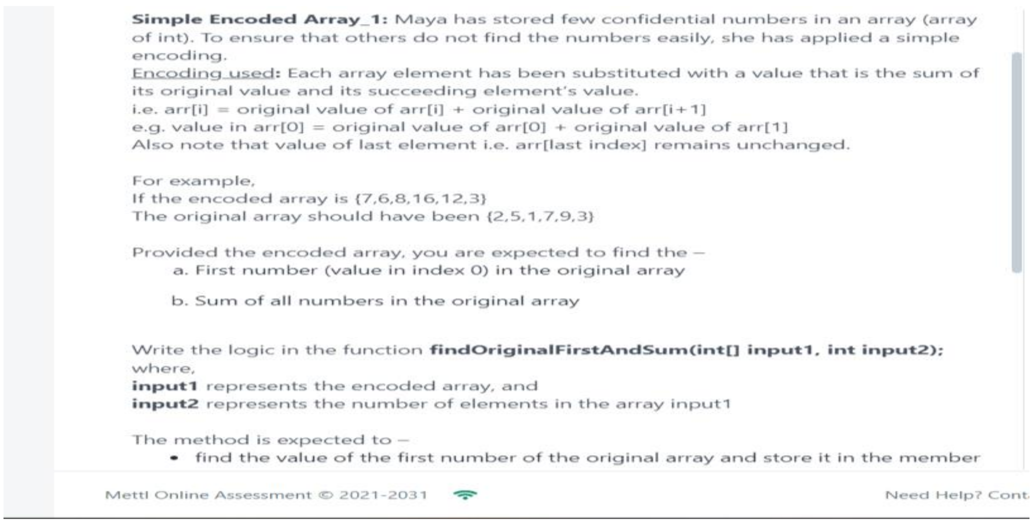
Assumption(s):
- The input strings will contain only numeric digits
- The input strings can be of any large lengths
- The lengths of the two input string need not be the same
- The input strings will represent only positive numbers

For example –
- If input strings are "1234" and "56", the output string should be "1290"
- If input strings are "56" and "1234", the output string should be "1290"
- If input strings are "123456732128989543219" and "987612673489652", the output string should be "123457719741663032871"

NOTE: In Java & C#, this logic can be easily implemented using BigInteger. However for the sake of enhancing your programming skills, you are recommended to solve this question without using BigInteger.

CODE 3:-

```
import java.io.*;
import java.util.*;
class UserMainCode
{
public int addNumberString(String input1,String input2)
{
int carry=0;
  if(input1.length()<input2.length())
  {
    String temp="";
   temp=input1;
   input1=input2;
   input2=temp;
  }
  int len1=input1.length();
  int len2=input2.length();
  String str="";
  int j=len2-1;
  for(int i=0;i<len1;i++)
  {
   int a=Character.getNumericValue(input1.charAt(len1-1-i));
   int b=0;
   if(j>=0)
   {
    b=Character.getNumericValue(input2.charAt(j));
    j--;
   }
   int sum=a+b+carry;
   carry=sum/10;
   int init=sum%10;
   str=Integer.toString(init)+str;
   if(i==len1-1 && carry>0)
   {
              str=Integer.toString(carry)+str;
    }
  }
  return str;
 }
}
```


Simple Encoded Array_1: Maya has stored few confidential numbers in an array (array of int). To ensure that others do not find the numbers easily, she has applied a simple encoding.
Encoding used: Each array element has been substituted with a value that is the sum of its original value and its succeeding element's value.
i.e. arr[i] = original value of arr[i] + original value of arr[i+1]
e.g. value in arr[0] = original value of arr[0] + original value of arr[1]
Also note that value of last element i.e. arr[last index] remains unchanged.

For example,
If the encoded array is {7,6,8,16,12,3}
The original array should have been {2,5,1,7,9,3}

Provided the encoded array, you are expected to find the –
a. First number (value in index 0) in the original array

b. Sum of all numbers in the original array

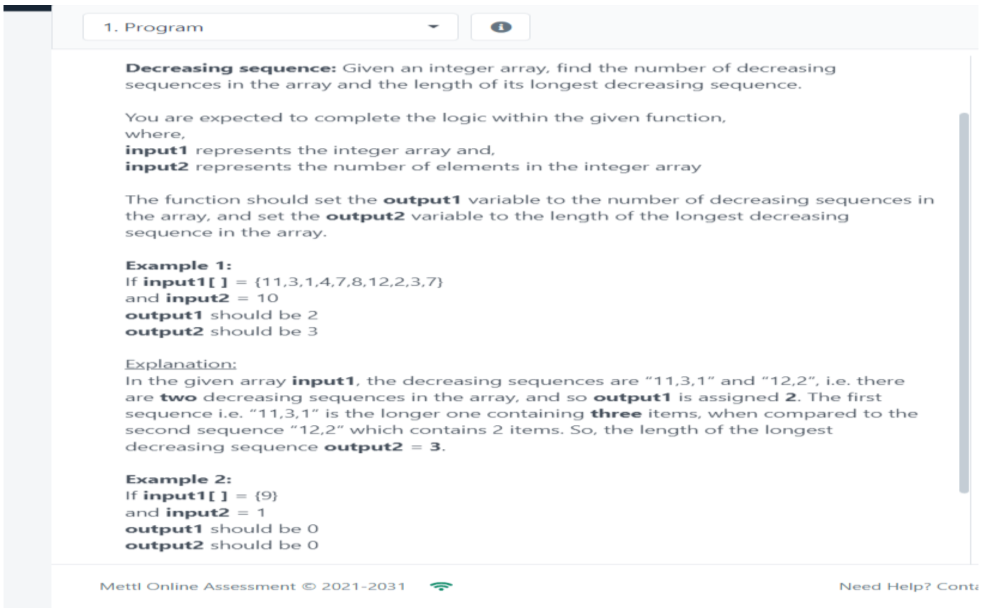
Write the logic in the function **findOriginalFirstAndSum(int[] input1, int input2);**
where,
input1 represents the encoded array, and
input2 represents the number of elements in the array input1

The method is expected to –
- find the value of the first number of the original array and store it in the member

Mettl Online Assessment © 2021-2031 Need Help? Cont

CODE 4:-

```
import java.io.*;
import java.util.*;
class UserMainCode
{
 public class Result
  {
    public final int output1;
    public final int output2;
public Result(int out1,int out2){
  output1=out1;
  output2=out2;
}}
  public Result findOriginalFirstAndSum(int[] input1,int input2){
int[] arr=new int[input2];
  arr[input2-1]=input1[input2-1];
  int sum=arr[input2-1];
  for(int i=input2-2;i>=0;i--)
        {
   arr[i]=input1[i]-arr[i+1];
   sum+=arr[i];
  }
         Result r1= new Result(arr[0],sum);
   return r1;
  }
    }
```



Decreasing sequence: Given an integer array, find the number of decreasing sequences in the array and the length of its longest decreasing sequence.

You are expected to complete the logic within the given function,
where,
input1 represents the integer array and,
input2 represents the number of elements in the integer array

The function should set the **output1** variable to the number of decreasing sequences in the array, and set the **output2** variable to the length of the longest decreasing sequence in the array.

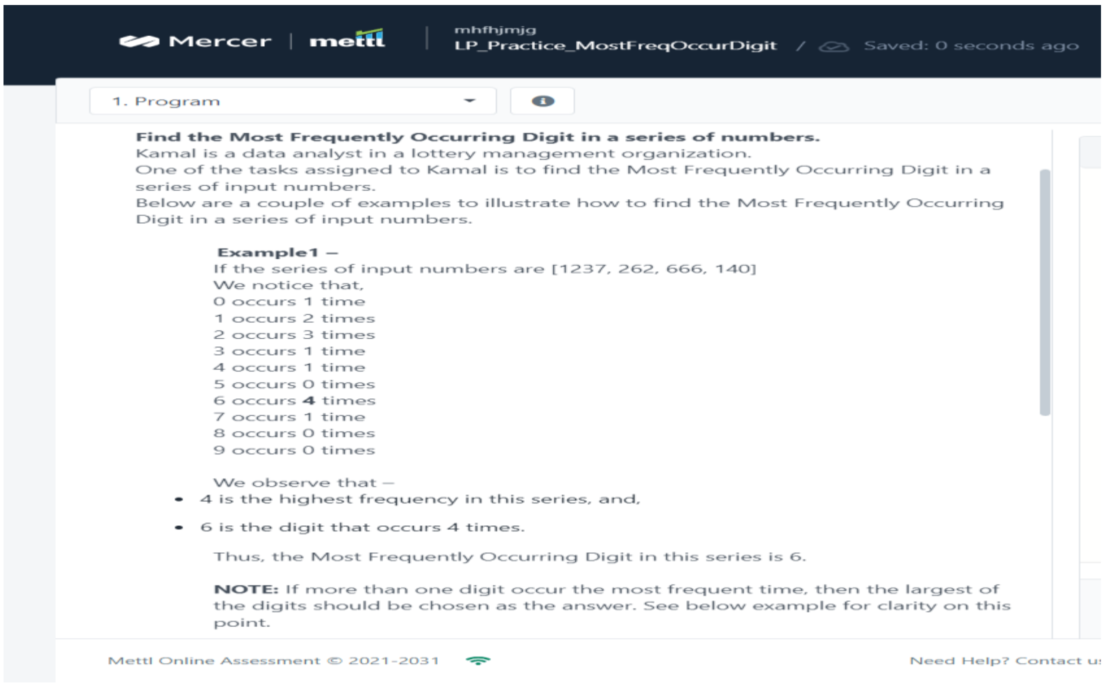
Example 1:
If **input1[ ]** = {11,3,1,4,7,8,12,2,3,7}
and **input2** = 10
output1 should be 2
output2 should be 3

Explanation:
In the given array **input1**, the decreasing sequences are "11,3,1" and "12,2", i.e. there are **two** decreasing sequences in the array, and so **output1** is assigned **2**. The first sequence i.e. "11,3,1" is the longer one containing **three** items, when compared to the second sequence "12,2" which contains 2 items. So, the length of the longest decreasing sequence **output2** = **3**.

Example 2:
If **input1[ ]** = {9}
and **input2** = 1
output1 should be 0
output2 should be 0

CODE 5:-

```
class UserMainCode
{
public class Result{
 public final int output1;
 public final int output2;
 public Result(int out1,int out2)
 {
  output1=out1;
  output2=out2;
 }
}
public Result decreasingSeq(int[] input1,int input2)
{
 int c1=0,c2=0,max=0;
for(int i=0;i<input2-1;i++)
        {
            if(input1[i]>input1[i+1])
            {
                c1++;
            }
            if((input1[i]<input1[i+1] && c1!=0) || ((i==input2-2) && c1!=0))
            {
                if(max<c1)
                {
                    max=c1;
                }
                c2++;
                c1=0;
            }
        }
        max=max+1;
        if(c2==0)
        {
            max=0;
        }
        if(input2==0)
        {
            max=0;
            c2=0;
        }
        Result r1= new Result(c2,max);
        return r1;
}
}
```


mhfhjmjg
LP_Practice_MostFreqOccurDigit / Saved: 0 seconds ago

1. Program

Find the Most Frequently Occurring Digit in a series of numbers.
Kamal is a data analyst in a lottery management organization.
One of the tasks assigned to Kamal is to find the Most Frequently Occurring Digit in a series of input numbers.
Below are a couple of examples to illustrate how to find the Most Frequently Occurring Digit in a series of input numbers.

Example1 –
If the series of input numbers are [1237, 262, 666, 140]
We notice that,
0 occurs 1 time
1 occurs 2 times
2 occurs 3 times
3 occurs 1 time
4 occurs 1 time
5 occurs 0 times
6 occurs **4** times
7 occurs 1 time
8 occurs 0 times
9 occurs 0 times

We observe that –
- 4 is the highest frequency in this series, and,
- 6 is the digit that occurs 4 times.

Thus, the Most Frequently Occurring Digit in this series is 6.

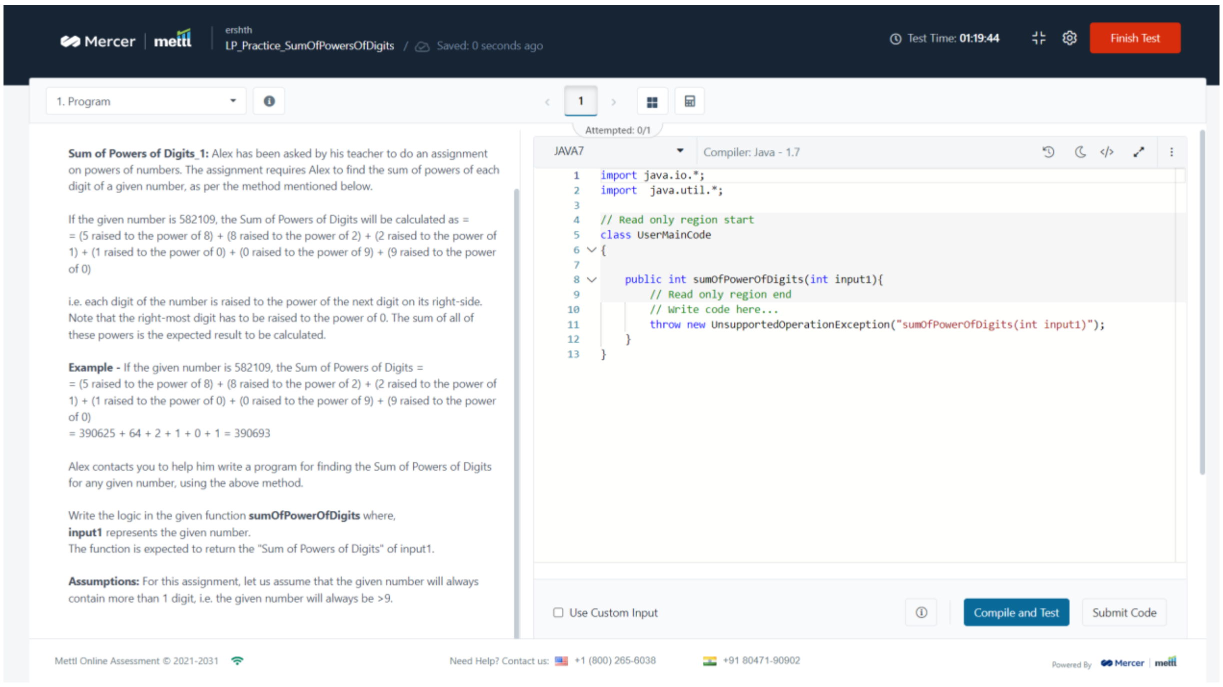
NOTE: If more than one digit occur the most frequent time, then the largest of the digits should be chosen as the answer. See below example for clarity on this point.

Mettl Online Assessment © 2021-2031 Need Help? Contact u

CODE 6:-

```
import java.io.*;
import  java.util.*;
class UserMainCode
{
public int mostFrequentlyOccurringDigit(int[] input1,int input2)
{
               int[] arr=new int[10];
  for(int i=0;i<input2;i++)
  {
    while(input1[i]!=0){
    int rem=input1[i]%10;
    arr[rem]++;
```

```
     input1[i]/=10;
   }
 }
 int max=0;
 int higest_occur_number=0;
 for(int i=0;i<10;i++)
 {
  if(arr[i]>=max)
  {max=arr[i];
     higest_occur_number=i;
  }
 }
 return higest_occur_number;
}}
```


Mercer | mettl
ershth
LP_Practice_SumOfPowersOfDigits / Saved: 0 seconds ago
Test Time: 01:19:44
Finish Test
1. Program
1
Attempted: 0/1
JAVA7
Compiler: Java - 1.7
Use Custom Input
Compile and Test
Submit Code
Mettl Online Assessment © 2021-2031
Need Help? Contact us: +1 (800) 265-6038 +91 80471-90902
Powered By Mercer | mettl

Sum of Powers of Digits_1: Alex has been asked by his teacher to do an assignment on powers of numbers. The assignment requires Alex to find the sum of powers of each digit of a given number, as per the method mentioned below.

If the given number is 582109, the Sum of Powers of Digits will be calculated as =
= (5 raised to the power of 8) + (8 raised to the power of 2) + (2 raised to the power of 1) + (1 raised to the power of 0) + (0 raised to the power of 9) + (9 raised to the power of 0)

i.e. each digit of the number is raised to the power of the next digit on its right-side. Note that the right-most digit has to be raised to the power of 0. The sum of all of these powers is the expected result to be calculated.

Example - If the given number is 582109, the Sum of Powers of Digits =
= (5 raised to the power of 8) + (8 raised to the power of 2) + (2 raised to the power of 1) + (1 raised to the power of 0) + (0 raised to the power of 9) + (9 raised to the power of 0)
= 390625 + 64 + 2 + 1 + 0 + 1 = 390693

Alex contacts you to help him write a program for finding the Sum of Powers of Digits for any given number, using the above method.

Write the logic in the given function **sumOfPowerOfDigits** where,
input1 represents the given number.
The function is expected to return the "Sum of Powers of Digits" of input1.

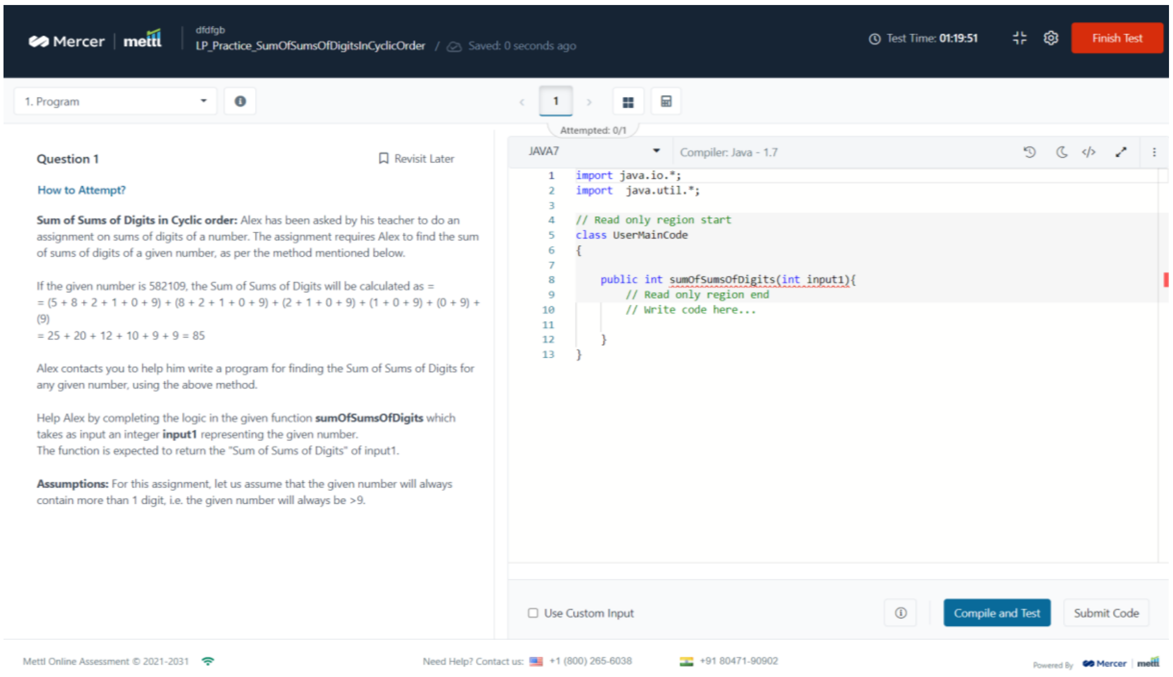
Assumptions: For this assignment, let us assume that the given number will always contain more than 1 digit, i.e. the given number will always be >9.

```java
import java.io.*;
import  java.util.*;

// Read only region start
class UserMainCode
{

    public int sumOfPowerOfDigits(int input1){
        // Read only region end
        // Write code here...
        throw new UnsupportedOperationException("sumOfPowerOfDigits(int input1)");
    }
}
```

CODE 7:-

```
import java.io.*;
import  java.util.*;

class UserMainCode{

public int sumOfPowerOfDigits(int input1){

double sum=0.0;
String str=Integer.toString(input1);
for(int i=0;i<str.length()-1;i++)  {

 int a=Character.getNumericValue(str.charAt(i));
 int b=Character.getNumericValue(str.charAt(i+1));
sum=sum + Math.pow(a, b);
```

```
 }
 return (int)sum+1;
}}
```



dfdfgb
LP_Practice_SumOfSumsOfDigitsInCyclicOrder / Saved: 0 seconds ago

Test Time: **01:19:51**

1. Program

Attempted: 0/1

Question 1

How to Attempt?

Sum of Sums of Digits in Cyclic order: Alex has been asked by his teacher to do an assignment on sums of digits of a number. The assignment requires Alex to find the sum of sums of digits of a given number, as per the method mentioned below.

If the given number is 582109, the Sum of Sums of Digits will be calculated as =
= (5 + 8 + 2 + 1 + 0 + 9) + (8 + 2 + 1 + 0 + 9) + (2 + 1 + 0 + 9) + (1 + 0 + 9) + (0 + 9) + (9)
= 25 + 20 + 12 + 10 + 9 + 9 = 85

Alex contacts you to help him write a program for finding the Sum of Sums of Digits for any given number, using the above method.

Help Alex by completing the logic in the given function **sumOfSumsOfDigits** which takes as input an integer **input1** representing the given number.
The function is expected to return the "Sum of Sums of Digits" of input1.

Assumptions: For this assignment, let us assume that the given number will always contain more than 1 digit, i.e. the given number will always be >9.

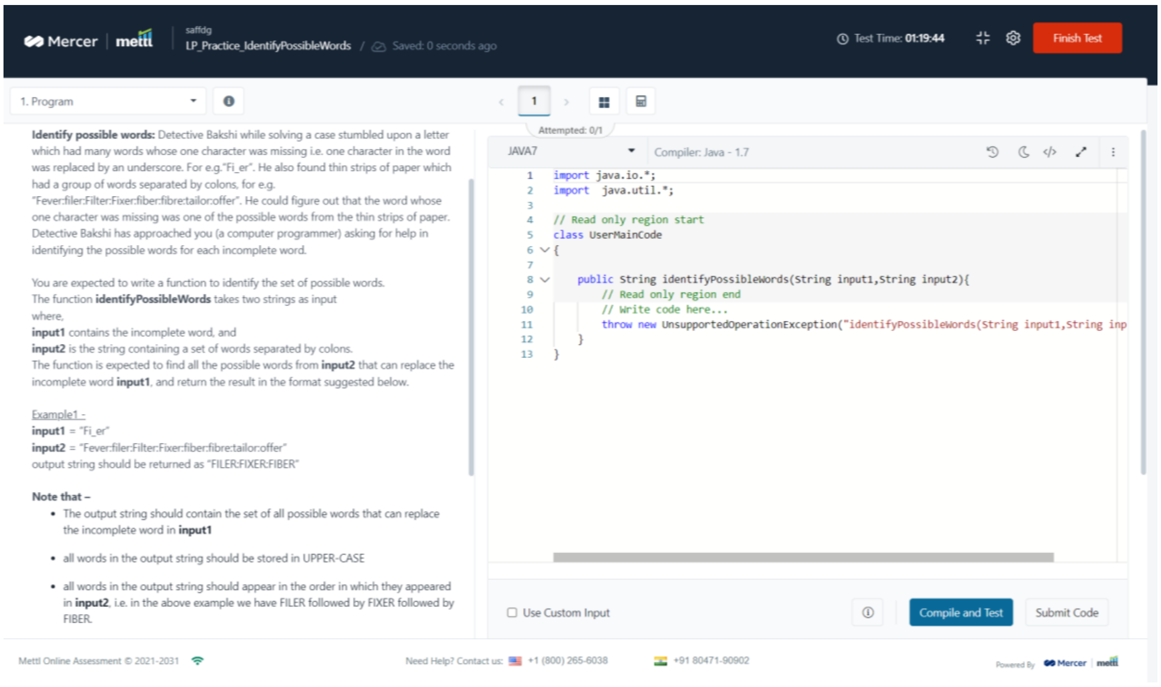
JAVA7 Compiler: Java - 1.7

```
import java.io.*;
import  java.util.*;

// Read only region start
class UserMainCode
{

    public int sumOfSumsOfDigits(int input1){
        // Read only region end
        // Write code here...

    }
}
```

Mettl Online Assessment © 2021-2031 Need Help? Contact us: +1 (800) 265-6038 +91 80471-90902

CODE 8:-

```
import java.io.*;

import  java.util.*;

class UserMainCode{

public int sumOfSumsOfDigits(int input1){

String str=Integer.toString(input1);

int sum=0;

for(int i=0;i<str.length();i++)  {
 for(int j=i;j<str.length();j++){

int num=Character.getNumericValue(str.charAt(j));

sum+=num;
}}

  return sum;

}}
```



LP_Practice_IdentifyPossibleWords

Identify possible words: Detective Bakshi while solving a case stumbled upon a letter which had many words whose one character was missing i.e. one character in the word was replaced by an underscore. For e.g."Fi_er". He also found thin strips of paper which had a group of words separated by colons, for e.g. "Fever:filer:Filter:Fixer:fiber:fibre:tailor:offer". He could figure out that the word whose one character was missing was one of the possible words from the thin strips of paper. Detective Bakshi has approached you (a computer programmer) asking for help in identifying the possible words for each incomplete word.

You are expected to write a function to identify the set of possible words.
The function **identifyPossibleWords** takes two strings as input
where,
input1 contains the incomplete word, and
input2 is the string containing a set of words separated by colons.
The function is expected to find all the possible words from **input2** that can replace the incomplete word **input1**, and return the result in the format suggested below.

Example1 -
input1 = "Fi_er"
input2 = "Fever:filer:Filter:Fixer:fiber:fibre:tailor:offer"
output string should be returned as "FILER:FIXER:FIBER"

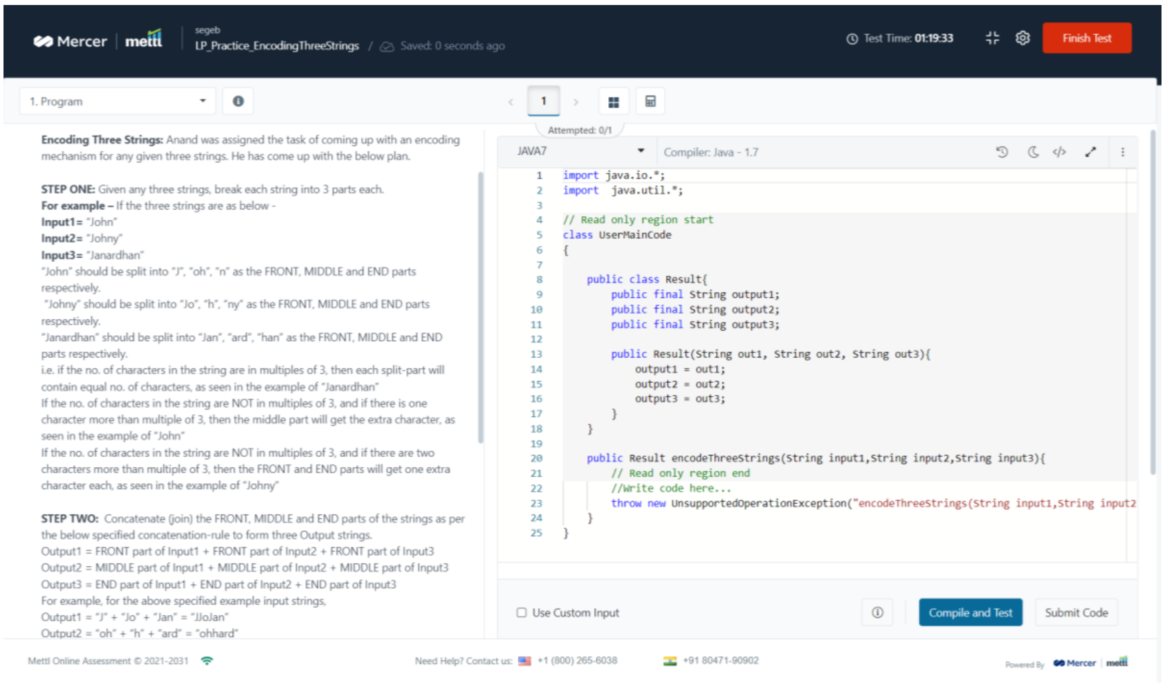
Note that –
- The output string should contain the set of all possible words that can replace the incomplete word in **input1**
- all words in the output string should be stored in UPPER-CASE
- all words in the output string should appear in the order in which they appeared in **input2**, i.e. in the above example we have FILER followed by FIXER followed by FIBER.

```
import java.io.*;
import  java.util.*;

// Read only region start
class UserMainCode
{

    public String identifyPossibleWords(String input1,String input2){
        // Read only region end
        // Write code here...
        throw new UnsupportedOperationException("identifyPossibleWords(String input1,String inp
    }
}
```

CODE 9:-

```
import java.io.*;
import java.util.*;
class UserMaincode
{
  public String indentifyPossibleWords(String input1, String input2)
{
String st[]=input2.split(":");
        String str="";
        for(int i=0;i<st.length;i++)
        {
            if(st[i].length()!=input1.length())
                continue;
            String test=input1;
            int x=test.indexOf("_");
            char ch=st[i].charAt(x);
            test=test.replace('_', ch);
            test=test.toUpperCase();
            st[i]=st[i].toUpperCase();
            if(st[i].equals(test))
            {
                if(str=="")
                {
                    str=str+test;
                }
                else
                {
                    str=str+":"+test;
                }
```

```
          }
       }
       if(str=="")
          str="ERROR-009";
       return str;
   }
   }
```



Encoding Three Strings: Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the below plan.

STEP ONE: Given any three strings, break each string into 3 parts each.
For example – If the three strings are as below -
Input1= "John"
Input2= "Johny"
Input3= "Janardhan"
"John" should be split into "J", "oh", "n" as the FRONT, MIDDLE and END parts respectively.
"Johny" should be split into "Jo", "h", "ny" as the FRONT, MIDDLE and END parts respectively.
"Janardhan" should be split into "Jan", "ard", "han" as the FRONT, MIDDLE and END parts respectively.
i.e. if the no. of characters in the string are in multiples of 3, then each split-part will contain equal no. of characters, as seen in the example of "Janardhan"
If the no. of characters in the string are NOT in multiples of 3, and if there is one character more than multiple of 3, then the middle part will get the extra character, as seen in the example of "John"
If the no. of characters in the string are NOT in multiples of 3, and if there are two characters more than multiple of 3, then the FRONT and END parts will get one extra character each, as seen in the example of "Johny"

STEP TWO: Concatenate (join) the FRONT, MIDDLE and END parts of the strings as per the below specified concatenation-rule to form three Output strings.
Output1 = FRONT part of Input1 + FRONT part of Input2 + FRONT part of Input3
Output2 = MIDDLE part of Input1 + MIDDLE part of Input2 + MIDDLE part of Input3
Output3 = END part of Input1 + END part of Input2 + END part of Input3
For example, for the above specified example input strings,
Output1 = "J" + "Jo" + "Jan" = "JJoJan"
Output2 = "oh" + "h" + "ard" = "ohhard"

```java
import java.io.*;
import  java.util.*;

// Read only region start
class UserMainCode
{

    public class Result{
        public final String output1;
        public final String output2;
        public final String output3;

        public Result(String out1, String out2, String out3){
            output1 = out1;
            output2 = out2;
            output3 = out3;
        }
    }

    public Result encodeThreeStrings(String input1,String input2,String input3){
        // Read only region end
        //Write code here...
        throw new UnsupportedOperationException("encodeThreeStrings(String input1,String input2
    }
}
```

Mettl Online Assessment © 2021-2031   Need Help? Contact us: +1 (800) 265-6038   +91 80471-90902   Powered By Mercer | mettl

CODE 10:-

```
String frnt1="",mid1="",end1="";
 String frnt2="",mid2="",end2="";
 String frnt3="",mid3="",end3="";
 String output1="",output2="",output3="";
 int len1=input1.length();
 int len2=input2.length();
        int len3=input3.length();
 if(len1==input1.length()){
 if(len1%3==0)
 {
  frnt1=input1.substring(0, (len1/3));
  mid1=input1.substring((len1/3), (2*(len1/3)));
  end1=input1.substring(2*(len1/3));
 }
 else if((len1-1)%3==0)
 {
  frnt1=input1.substring(0, (len1/3));
  mid1=input1.substring((len1/3), ((2*(len1/3))+1));
  end1=input1.substring(((2*(len1/3))+1));
}
else if((len1-2)%3==0)
{
 frnt1=input1.substring(0, ((len1/3)+1));
 mid1=input1.substring(((len1/3)+1), ((2*(len1/3))+1));
 end1=input1.substring(((2*(len1/3))+1));
}
}
 if(len2==input2.length()){
if(len2%3==0)
{
 frnt2=input2.substring(0, (len2/3));
 mid2=input2.substring((len2/3), (2*(len2/3)));
 end2=input2.substring(2*(len2/3));
}
else if((len2-1)%3==0)
{
 frnt2=input2.substring(0, (len2/3));
 mid2=input2.substring((len2/3), ((2*(len2/3))+1));
 end2=input2.substring(((2*(len2/3))+1));
}
else if((len2-2)%3==0)
{
 frnt2=input2.substring(0, ((len2/3)+1));
 mid2=input2.substring(((len2/3)+1), ((2*(len2/3))+1));
 end2=input2.substring(((2*(len2/3))+1));
}
}
 if(len3==input3.length()){
if(len3%3==0)
{
 frnt3=input3.substring(0, (len3/3));
 mid3=input3.substring((len3/3), (2*(len3/3)));
 end3=input3.substring(2*(len3/3));
}
else if((len3-1)%3==0)
{
 frnt3=input3.substring(0, (len3/3));
 mid3=input3.substring((len3/3), ((2*(len3/3))+1));
 end3=input3.substring(((2*(len3/3))+1));
}
else if((len3-2)%3==0)
{
 frnt3=input3.substring(0, ((len3/3)+1));
 mid3=input3.substring(((len3/3)+1), ((2*(len3/3))+1));
 end3=input3.substring(((2*(len3/3))+1));
}
}
output1=frnt1+frnt2+frnt3;
output2=mid1+mid2+mid3;
output3=end1+end2+end3;
```

```
 System.out.println(output3);
 output3=changeCase(output3);
 Result rs=new Result(output1,output2,output3);
 return rs;
   }
public static String changeCase(String str)
{
 StringBuffer newS = new StringBuffer(str);
 for(int i=0;i<str.length();i++)
 {
  Character c=str.charAt(i);
  if(Character.isLowerCase(c))
  {
   newS.replace(i, i+1, Character.toUpperCase(c)+"");
  }
  else
  {
   newS.replace(i, i+1, Character.toLowerCase(c)+"");
  }
 }
 str=newS.toString();
 return str;
}
```