tests.mettl.com/test-window/1wavwa4c1s#/testWindow/0/0/2

mettl

TalentNext_2020_Le

ection 1 of 2 Section 1

Question # 1

Revisit

How to attempt?

Question

Find Key:

You are provided with 3 numbers : **input1, input2 and input3.**

Each of these are four digit numbers within the range >=1000 and <=9999.

i.e.

1000 <= input1 <= 9999

1000 <= input2 <= 9999

1000 <= input3 <= 9999

You are expected to find the Key using the below formula -

Key = (Thousands digit of input1 x Hundreds digit of input2) + (Smallest digit of input3)

For e. g. if input1 = 3521, input2=2452, input3=1352, then Key = (3 x 4) + 1 = 13

Assuming that the 3 numbers are passed to the given function, Complete the function to find and return the Key.

mettl

Thanesshwaran | Support

```
import java.util.*;

public class HelloWorld {
    public static void main(String[] args) {
       Scanner sc=new Scanner(System.in);
       int a=sc.nextInt();
       int b=sc.nextInt();
       int c=sc.nextInt();
       int th=(a%10000)/1000;
       int hu=(b%1000)/100;
       int x=9;
       int y=0;
       for(int i=0;i<4;i++)
       {
          y=c%10; // 2
          c=c/10;    // 1 3 5
          if (y<x) //2<10
          {
            x=y;
          }
       }
       System.out.println((th*hu)+x);
    }
}
```

Online Test Window - Google Chrome

tests.mettl.com/test-window/1wavwa4c1s#/testWindow/0/0/2

mettl

TalentNext_2020_LearningPhase_Mile1_LPs

Section 1 of 2 Section 1

1

Question # 1

Revisit

How to attempt?

Question

Find Key:

You are provided with 3 numbers : **input1, input2 and input3.**

Each of these are four digit numbers within the range >=1000 and <=9999.

i.e.

1000 <= input1 <= 9999

1000 <= input2 <= 9999

1000 <= input3 <= 9999

You are expected to find the Key using the below formula -

Key = (Hundreds digit of input1 x Tens digit of input2) - (Largest digit of input3)

For e. g. if input1 = 3521, input2=2452, input3=1352, then Key = (5 x 5) - 5 = 20

Assuming that the 3 numbers are passed to the given function, Complete the function to find and return the Key

Language: C

Compiler gcc 5.4.0

```
1  #include<stdio.h>
2  #include<string.h>
3  // Read only region start
4
5  int findKey(int input1,int input2,
6  {
7      // Read only region end
8      // Write code here
9
10 }
```

mettl

Bhuvaneshwaran | Support +1-650-924-9221 +91-82878-03040

Type here to search

```
import java.util.*;

public class HelloWorld {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();
        int hu=(a%1000)/100;
        int te=(b%100)/10;
        int x=0;
        int y=0;
        while(c!=0)
        {
            y=c%10;
            c=c/10;
            if (y>x)
            {
              x=y;
            }
        }
        System.out.println((hu*te)-x);
    }
}
```

mettl

TalentNext_2020_Le

Section 1 of 2 Section 1

Question # 1

Revisit

How to attempt?

Question :

Find Key:

You are provided with 3 numbers : **input1, input2 and input3.**

Each of these are four digit numbers within the range >=1000 and <=9999.

i.e.

1000 <= input1 <= 9999

1000 <= input2 <= 9999

1000 <= input3 <= 9999

You are expected to find the Key using the below formula -

Key = [largest digit in the thousands place of all three numbers] [largest digit in the hundreds place of all three numbers] [largest digit in the tens place of all three numbers] [largest digit in the units place of all three numbers]

For e..g. if input1 = 3521, input2=2452, input3=1352, then Key = [3] [5] [5] [2] = 3552

Assuming that the 3 numbers are passed to the given function, Complete the function to find and return the Key.

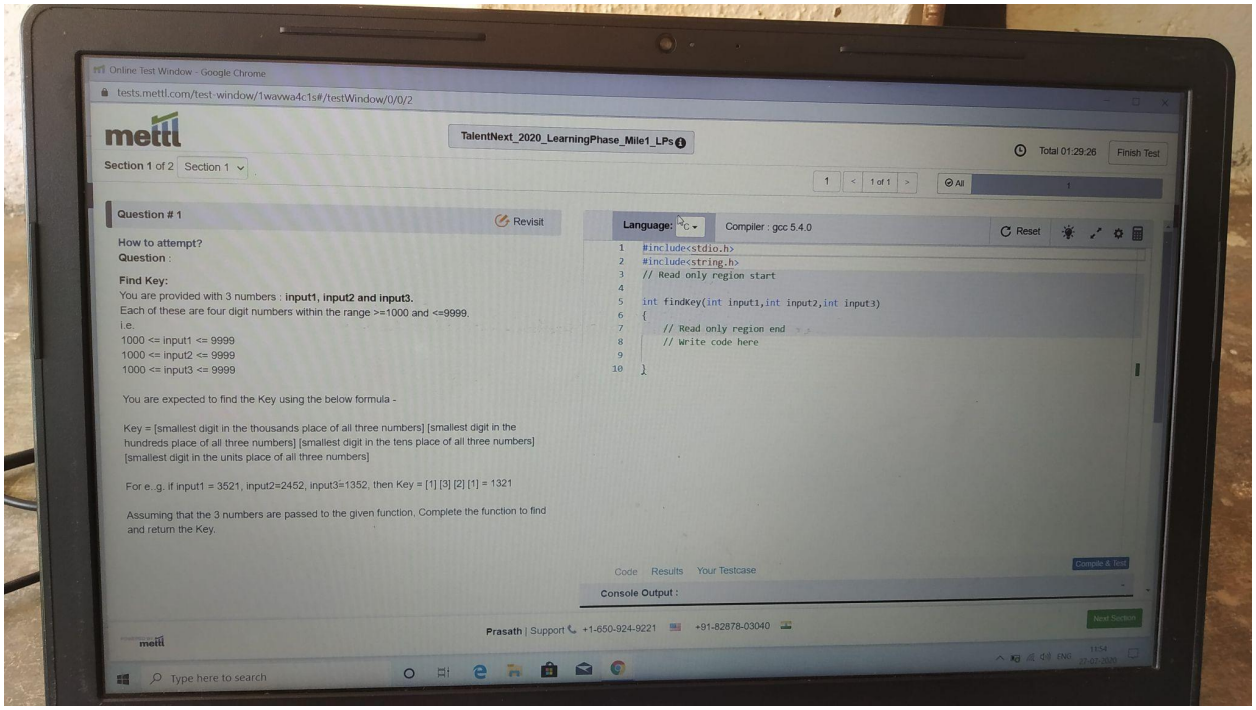
mettl

Devipriya | Support

Type here to search

```
import java.util.*;

public class HelloWorld {
    public static void main(String[] args) {
       Scanner sc=new Scanner(System.in);
       int a=sc.nextInt();   //input1
       int b=sc.nextInt();   //input2
       int c=sc.nextInt();   //input3
       int th=10000;         //10000
       int hu=th/10;         //1000
       int large;            //
       String s="";          //string
       while(a!=0)
       {
          large=0;
          int m=(a%th)/hu;   //(a%10)/=3
          if(large<m)        // 0<3
           large=m;          //large=3
          int n=(b%th)/hu;   //2
          if(large<n)        //3<2
           large=n;
          int o=(c%th)/hu;   //1
          if(large<o)        //3<1
           large=o;
          s=s+""+large;      //3
          th=th/10;          //10000/10=1000
          hu=hu/10;          //1000/10=100
          if (hu==0)
            break;
       }
       System.out.println(s);
    }
}
```




```
import java.util.*;

public class HelloWorld {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      int a=sc.nextInt();   //input1
      int b=sc.nextInt();   //input2
      int c=sc.nextInt();   //input3
      int th=10000;         //10000
      int hu=th/10;         //1000
      int small;            //
      String s="";          //string
      while(a!=0)
      {
         small=10;
         int m=(a%th)/hu;   //(a%10)/=3
         if(small>m)        // 0<3
```

```
            small=m;          //large=3
           int n=(b%th)/hu;   //2
           if(small>n)        //3<2
            small=n;
           int o=(c%th)/hu;   //1
           if(small>o)        //3<1
            small=o;
           s=s+""+small;      //3
           th=th/10;          //10000/10=1000
           hu=hu/10;          //1000/10=100
           if (hu==0)
             break;
        }
        System.out.println(s);
    }
}
```

Section 1 - Google Chrome

tests.mettl.com

mettl

TalentNext_2020_Learnin

Section 2 of 2 Section 2

writing a function to generate the password.

The scenario is as below -
Five numbers are available with the kids.
These numbers are either stable or unstable.
A number is **stable** if each of its digit occur the same number of times, i.e. the frequency of each digit in the number is the same. For e.g. 2277, 4004, 11, 23, 583835, 1010 are examples of stable numbers.
Similarly, A number is **unstable** if the frequency of each digit in the number is NOT the same. For e.g. 221, 4314, 101, 233, 58135, 101 are examples of unstable numbers..

The password can be found as below -
i.e. **password = Maximum of all stable numbers - Minimum of all Unstable numbers**

Assuming that the five numbers are passed to a function as input1, input2, input3, input4 and input5, complete the function to find and return the password.
For Example:
If input1 = 12, input2 = 1313, input3 = 122, input4 = 678, and input5 = 898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the password should be = **Maximum of all stable numbers - Minimum of all Unstable numbers = 1313 - 122 = 1191**

mettl

Veeralakshmi | Support +1

```
import java.util.*;

public class HelloWorld {
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      int a[]=new int[5];
      int stable=0;
      int max=0;
      int min=0;
      for(int i=0;i<5;i++)
      {
         int val=sc.nextInt();
         a[i]=val;
      }
      for(int i=0;i<5;i++)
      {
         int x=a[i];
         int x1=(x%10)/1;
         String v1=""+x;
         //System.out.println(v1);
         int count=0;
         int th=1;
         for(int j=0;j<v1.length();j++)
             th=th*10;
         int hu=th/10;
         while(hu!=0)
         {
           int single=(x%th)/hu;
           if (single==x1)
           {
             count=count+1;
           }
           th=th/10;
           hu=hu/10;
```

```
}

int count1;
th=1;
for(int j=0;j<v1.length();j++)
    th=th*10;

hu=th/10;
int ten=th;
int hun=hu;
count1=0;
while(hu!=0)//for(int j=0;j<v1.length();j++)
{
      count1=0;
      int ht=ten;
      int uh=hun;
      int single=(x%th)/hu;
      while(uh!=0)//for(int k=0;k<v1.length();j++)
      {
        int spec=(x%ht)/uh;
        if(single==spec)
        {
          count1=count1+1;
        }
        ht=ht/10;
        uh=uh/10;
        if(uh==0)
          break;

      }
      if(count==count1)
      {
         stable=1;
      }
      else
```

```
            {
               stable=0;
               break;
            }
            th=th/10;
            hu=hu/10;


        }
        if(stable==1)
        {

          if(x>max)
          {
            max=x;
          }

        }
        if(stable==0)
        {
          if (min==0)
          {
             min=x;
          }
          else if(x<min)
          {
             min=x;
          }
        }
      }
      System.out.println(max-min);

   }
}
```

mettl

TalentNext_2020_Learning

Section 2 of 2 Section 2

Question # 1

Revisit

How to attempt?

Question :

Find Password:

Detective Buckshee Junior has been approached by the shantiniketan kids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee Junior realises that he will need a programmer's support. He contacts you and requests your help. Please help the detective by writing a function to generate the password.

The scenario is as below -

Five numbers are available with the kids.

These numbers are either stable or unstable.

A number is **stable** if each of its digit occur the same number of times, i.e. the frequency of each digit in the number is the same. For e.g. 2277, 4004, 11, 23, 583835, 1010 are examples of stable numbers..

Similarly, A number is **unstable** if the frequency of each digit in the number is NOT the same. For e.g. 221, 4314, 101, 233, 58135, 101 are examples of unstable numbers..

The password can be found as below -

i.e. **password = Maximum of all Unstable numbers + Minimum of all Unstable numbers**

Assuming that the five numbers are passed to a function as input1, input2, input3, input4 and input5, complete the function to find and return the password.

For Example:

If input1 = 12, input2 = 1313, input3 = 122, input4 = 678, and input5 = 898,

we see that there are

mettl

Devipriya | Support +1-65

Type here to search

```
import java.util.*;

public class HelloWorld {
   public static void main(String[] args)
   {
      Scanner sc=new Scanner(System.in);
      int a[]=new int[5];
      int stable=0;
      int max=0;
      int min1=0;
      int max1=0;
      int min=0;
      for(int i=0;i<5;i++)
      {
        int val=sc.nextInt();
        a[i]=val;
      }
      for(int i=0;i<5;i++)
      {
        int x=a[i];
        int x1=(x%10)/1;
        String v1=""+x;
        //System.out.println(v1);
        int count=0;
        int th=1;
        for(int j=0;j<v1.length();j++)
            th=th*10;
        int hu=th/10;
        while(hu!=0)
        {
          int single=(x%th)/hu;
          if (single==x1)
          {
            count=count+1;
          }
          th=th/10;
          hu=hu/10;
        }

        int count1;
        th=1;
        for(int j=0;j<v1.length();j++)
            th=th*10;
```

```
hu=th/10;
int ten=th;
int hun=hu;
count1=0;
while(hu!=0)//for(int j=0;j<v1.length();j++)
{
      count1=0;
      int ht=ten;
      int uh=hun;
      int single=(x%th)/hu;
      while(uh!=0)//for(int k=0;k<v1.length();j++)
      {
        int spec=(x%ht)/uh;
        if(single==spec)
        {
          count1=count1+1;
        }
        ht=ht/10;
        uh=uh/10;
        if(uh==0)
          break;

      }
      if(count==count1)
      {
         stable=1;
      }
      else
      {
         stable=0;
         break;
      }
      th=th/10;
      hu=hu/10;


}
if(stable==1)
{

  if(x>max)
  {
    max=x;
  }
```

```
            if(min==0)
            {
             min=x;
            }
            else if(x<min)
            {
              min=x;
            }

          }
          if(stable==0)
          {
            if (min1==0)
            {
               min1=x;
            }
            else if(x<min1)
            {
               min1=x;
            }
            if(max1==0)
            {
              max1=x;
            }
            else if(x>max1)
            {
              max1=x;
            }
          }
        }
        System.out.println(max1-min1);

    }
}
```

mettl

TalentNext_2020_Le

Section 2 of 2 Section 2

Detective Buckshee Junior has been approached by the shantiniketan kids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee Junior realises that he will need a programmer's support. He contacts you and requests your help. Please help the detective by writing a function to generate the password.

The scenario is as below -
Five numbers are available with the kids.
These numbers are either stable or unstable.
A number is **stable** if each of its digit occur the same number of times, i.e. the frequency of each digit in the number is the same. For e.g. 2277, 4004, 11, 23, 583835, 1010 are examples of stable numbers..
Similarly, A number is **unstable** if the frequency of each digit in the number is NOT the same. For e.g. 221, 4314, 101, 233, 58135, 101 are examples of unstable numbers..

The password can be found as below -
i.e. **password = Maximum of all Unstable numbers + Minimum of all Unstable numbers**

Assuming that the five numbers are passed to a function as input1, input2, input3, input4 and input5, complete the function to find and return the password.
For Example:
If input1 = 12, input2 = 1313, input3 = 122, input4 = 678, and input5 = 898,
we see that there are
THREE stable numbers i.e. 12, 1313 and 678 and
TWO unstable numbers i.e. 122 and 898
So, the password should be = **Maximum of all Unstable numbers + Minimum of all Unstable numbers = 898 + 122 = 1020**

mettl

Devipriya | Support +1-

Type here to search

```
import java.util.*;

public class HelloWorld {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int a[]=new int[5];
        int stable=0;
        int max=0;
        int min1=0;
        int max1=0;
        int min=0;
        for(int i=0;i<5;i++)
        {
           int val=sc.nextInt();
           a[i]=val;
        }
        for(int i=0;i<5;i++)
        {
           int x=a[i];
           int x1=(x%10)/1;
           String v1=""+x;
           //System.out.println(v1);
           int count=0;
           int th=1;
           for(int j=0;j<v1.length();j++)
                th=th*10;
           int hu=th/10;
           while(hu!=0)
           {
             int single=(x%th)/hu;
             if (single==x1)
             {
               count=count+1;
             }
             th=th/10;
             hu=hu/10;
           }

           int count1;
```

```
th=1;
for(int j=0;j<v1.length();j++)
    th=th*10;

hu=th/10;
int ten=th;
int hun=hu;
count1=0;
while(hu!=0)//for(int j=0;j<v1.length();j++)
{
      count1=0;
      int ht=ten;
      int uh=hun;
      int single=(x%th)/hu;
      while(uh!=0)//for(int k=0;k<v1.length();j++)
      {
        int spec=(x%ht)/uh;
        if(single==spec)
        {
          count1=count1+1;
        }
        ht=ht/10;
        uh=uh/10;
        if(uh==0)
          break;

      }
      if(count==count1)
      {
         stable=1;
      }
      else
      {
         stable=0;
         break;
      }
      th=th/10;
      hu=hu/10;
```

```
    }
    if(stable==1)
    {

      if(x>max)
      {
        max=x;
      }
      if(min==0)
      {
       min=x;
      }
      else if(x<min)
      {
        min=x;
      }

    }
    if(stable==0)
    {
      if (min1==0)
      {
         min1=x;
      }
      else if(x<min1)
      {
         min1=x;
      }
      if(max1==0)
      {
        max1=x;
      }
      else if(x>max1)
      {
        max1=x;
      }
    }
}
System.out.println(max1+min1);
     }}
```

writing a function to generate the password.

The scenario is as below -

Five numbers are available with the kids.

These numbers are either stable or unstable.

A number is **stable** if each of its digit occur the same number of times, i.e. the frequency of each digit in the number is the same. For e.g. 2277, 4004, 11, 23, 583835, 1010 are examples of stable numbers.

Similarly, A number is **unstable** if the frequency of each digit in the number is NOT the same. For e.g. 221, 4314, 101, 233, 58135, 101 are examples of unstable numbers..

The password can be found as below -

i.e. **password = Maximum of all stable numbers + Minimum of all stable numbers**

Assuming that the five numbers are passed to a function as input1, input2, input3, input4 and input5, complete the function to find and return the password.

For Example:

If input1 = 12, input2 = 1313, input3 = 122, input4 = 678, and input5 = 898,

stable numbers are 12, 1313 and 678

unstable numbers are 122 and 898

So, the password should be = **Maximum of all stable numbers + Minimum of all stable numbers = 1313 + 12 = 1325**

```
import java.util.*;

public class HelloWorld {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int a[]=new int[5];
        int stable=0;
        int max=0;
        int min1=0;
        int max1=0;
        int min=0;
        for(int i=0;i<5;i++)
        {
           int val=sc.nextInt();
           a[i]=val;
        }
        for(int i=0;i<5;i++)
        {
           int x=a[i];
           int x1=(x%10)/1;
           String v1=""+x;
           //System.out.println(v1);
           int count=0;
           int th=1;
           for(int j=0;j<v1.length();j++)
                th=th*10;
           int hu=th/10;
           while(hu!=0)
           {
             int single=(x%th)/hu;
             if (single==x1)
             {
               count=count+1;
             }
             th=th/10;
             hu=hu/10;
           }

           int count1;
```

```
th=1;
for(int j=0;j<v1.length();j++)
    th=th*10;

hu=th/10;
int ten=th;
int hun=hu;
count1=0;
while(hu!=0)//for(int j=0;j<v1.length();j++)
{
      count1=0;
      int ht=ten;
      int uh=hun;
      int single=(x%th)/hu;
      while(uh!=0)//for(int k=0;k<v1.length();j++)
      {
        int spec=(x%ht)/uh;
        if(single==spec)
        {
          count1=count1+1;
        }
        ht=ht/10;
        uh=uh/10;
        if(uh==0)
          break;

      }
      if(count==count1)
      {
         stable=1;
      }
      else
      {
         stable=0;
         break;
      }
      th=th/10;
      hu=hu/10;
```

```
    }
    if(stable==1)
    {

      if(x>max)
      {
        max=x;
      }
      if(min==0)
      {
       min=x;
      }
      else if(x<min)
      {
        min=x;
      }

    }
    if(stable==0)
    {
      if (min1==0)
      {
         min1=x;
      }
      else if(x<min1)
      {
         min1=x;
      }
      if(max1==0)
      {
        max1=x;
      }
      else if(x>max1)
      {
        max1=x;
      }
    }
 }
 System.out.println(max+min);
} }
```