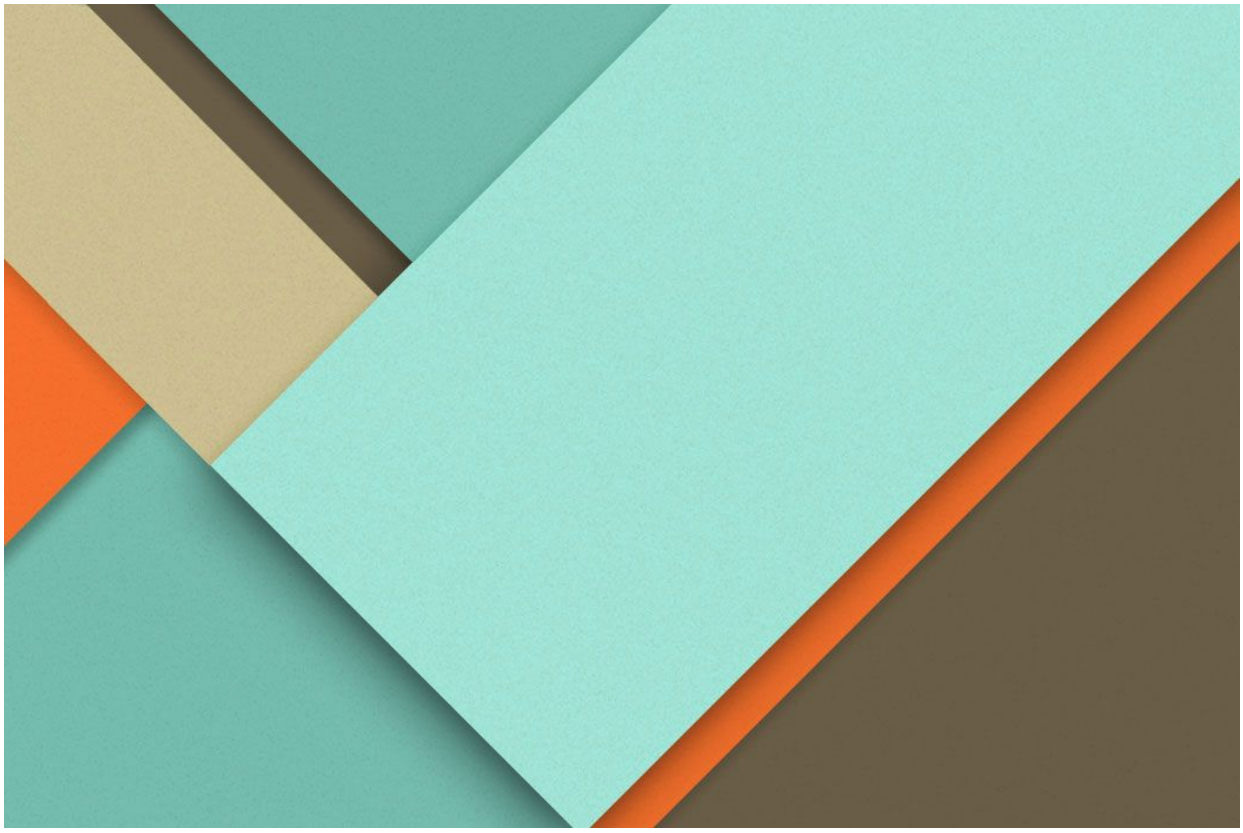

Coding Assignment

12.04.2017

Chandra Sekhar B

Madhapur,
Hiderabad -500 010

01. Sum of the Digits.

Define a method which returns the sum of digits of the given two digit number.

Write the method with the following specifications:

Name of method **getSumOfDigits()** // which accepts an integer value as argument and return the sum of it's digits.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *getSumOfdigits()* is determined by the following rules:

- if the given value is in between 10 and 99, return sum of it's digits. Example: if x = 34, return 7
- if the given value is negative, return -3
- if the given value is greater than 99, return -2
- if the given value is in between 0 and 9, return -1

Read the steps below carefully before you start

1. Use the skeleton code provided ***(ECC_01_SumOfDigits.java)***
2. In the Template file, add your code in the placeholder - "ADD YOUR CODE HERE"
3. To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc
4. Compile your code
5. Check the output and upload the source file i.e., .java file

Template:

```
public class ECC_01_SumOfDigits {
        public static void main(String[] args) {
        int num = 67;
        System.out.println(getSumOfDigits(num));
        }

        public static int getSumOfDigits(int num)    {
        // ADD YOUR CODE HERE
        }
}
```

02. Digit Checker

Define a method which returns the difference of digits of the given two digit number.

Note: You should substract the units position value from tens positon value, the return value may be negative.

Write the method with the following specifications

Name of method *getDiffOfDigits()* // which accepts an integer value as argument and return the difference of it's digits.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *getDiffOfdigits()* is determined by the following rules

if the given value is in between 10 and 99, return difference of it's digits.

Example: if x = 83, 8 - 3 return 5. if x = 38, 3 - 8 return -5.

if the given value is negative, return -3

if the given value is greater than 99, return -2

if the given value is in between 0 and 9, return -1

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_02_DigitChecker.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_02_DigitChecker {
        public static void main(String[] args) {
        int num = 83;
        System.out.println(getDiffOfDigits(num));
        }
        public static int getDiffOfDigits(int num) {
   // ADD YOUR CODE HERE
        }
}
```

03. Multiple Of 100:

Define a method which returns the next multiple of 100 for the given number.

Write the method with the following specifications

Name of method *getNextMultipleOf100()* // which accepts an integer value as argument and return the next multiple of 100.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *getNextMultipleOf100()* is determined by the following rules

if the given value is negative or zero, return -1

if the given value is positive, return the next multiple of the given number.

Example: if x = 123, return 200.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_03_MultipleOf100.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_03_MultipleOf100 {
        public static void main(String[] args) {
        int num = 123;
        System.out.println(getNextMultipleOf100(num));
        }

        public static int getNextMultipleOf100(int num) {
        // ADD YOUR CODE HERE
        }
}
```

04. Is Palindrome

Define a method which returns the 1 if the given three digit number is palindrome, in other case return 0.

Write the method with the following specifications

Name of method *isPalindrome()* // which accepts an integer value as argument and return true if the given number is palindrome, else retrun false.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *isPalindrome()* is determined by the following rules

if the given number is an three digit number, retun 1 if the number is palindrome, else return 0.

Example: if x = 232, return 1. if x = 345, return 0

if the given number is negative or zero, return -1

if the given number is not an three digit number, return -2

Read the steps below carefully before you start

1. **use skeleton code provided *(ECC_04_Palindrome.java)***
2. **add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_03_MultipleOf100 {
        public static void main(String[] args) {
        int num = 123;
        System.out.println(getNextMultipleOf100(num));
        }

        public static int getNextMultipleOf100(int num) {
        // ADD YOUR CODE HERE
        }
}
```

05. Even Finder :

Define a method which returns the 1 if the given number is even, in other case return 0
Write the method with the following specifications
Name of method *isEven()* // which accepts an integer value as argument and return 1 if the given number is even, else retrun 0.
Arguments: one argument of type integer
Return Type: an integer value
Specifications: The value returned by the method *isEven()* is determined by the following rules
if the given number is an even number, return 1 else return 0. Example if x = 22, return 1. if x = 35, return 0
if the given number is negative or zero, return -1

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_05_EvenFinder.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

```
class ECC_05_EvenFinder {
        public static void main(String[] args) {
        int num = 24;
        System.out.println(isEven(num));
        }

        public static int isEven(int num) {
        // ADD YOUR CODE HERE
        }
}
```

06. GratestNumber:

Define a method which returns the greatest number among two numbers.
Write the method with the following specifications
Name of method *getGreatest()* / / which accepts two integer values as argument and return the greatest value.
Arguments: two argument of type integer
Return Type: an integer value
Specifications: The value returned by the method *getGreatest()* is determined by the following rules
if any of the given numbers are negative, return -1.
if any of the given numbers are zero, return -2.
if the given numbers are positive, return the greatest.
Read the steps below carefully before you start

1. **use skeleton code provided *(ECC_06_GreatestNumber.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_06_GreatestNumber {
        public static void main(String[] args) {
        int n1 = 10;
        int n2 = 45;
        System.out.println(getGreatest(n1, n2));
        }

        public static int getGreatest(int num1, int num2) {
        // ADD YOUR CODE HERE
        }
}
```

07. Least Number:

Define a method which returns the least number among two numbers.
Write the method with the following specifications
Name of method *getLeastNum()* // which accepts two integer values as argument and return the least value. Arguments: two argument of type integer Return Type: an integer value Specifications: The value returned by the method *getLeastNum()* is determined by the following rules
if any of the given numbers are negative, return -1.
if any of the given numbers are zero, return -2.
if the given numbers are positive, return the least number.
Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_07_LeastNumber.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_07_LeastNumber {

        public static void main(String[] args) {

        int n1 = 10;

        int n2 = 45;

        System.out.println(getLeastNum(n1, n2));

        }

      public static int getLeastNum(int num1, int num2) {

        // ADD YOUR CODE HERE

        }

}
```

08. OddRounder:

Define a method which returns the number itself if it is an even number, if the number is odd then return the next multiple of 10.

Write the method with the following specifications

Name of method *oddRounder()* // which accepts an integer value as argument and return the same value if it is an even number, if the value is odd then return the next multiple of 10. Example if x = 24 then return 24, if x = 25 then return 30.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *oddRounder()* is determined by the following rules

if any of the given number is negative, return -1.

if any of the given number is zero, return -2.

if the given number is even, return the same number.

if the given number is odd, return the next multiple of 10.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_08_OddRounder.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
class ECC_08_OddRounder {
        public static void main(String[] args) {
        int num = 3;
        System.out.println(oddRounder(num));
        }

        public static int oddRounder(int num) {
        // ADD YOUR CODE HERE
        }

}
```

09.SignFinder:

Define a method which returns the 1 if the given number is positive, return -1 if the given number is negative, return 0 if the given number is 0.

Write the method with the following specifications

Name of method *findSign()* // which accepts an integer value as argument and return 1 if the argument value is positive, return -1 in case of negative value, return 0 if the argument value is 0.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *findSign()* is determined by the following rules:

if any of the given number is positive, return 1.

if any of the given number is negative, return -1.

if any of the given number is zero, return 0.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_09_SignFinder.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
class ECC_09_SignFinder {

        public static void main(String[] args) {

        int num = 34;

        System.out.println(findSign(num));

        }


        public static int findSign(int num) {

        // ADD YOUR CODE HERE

        }

}
```

10. Even Or Odd

Define a method which returns the string as "Even" if the given number is an even number, return "Odd" if the given number is an odd number, return string as "Invalid Input" if the given number is lessthan or equal to 0.

Write the method with the following specifications

Name of method *isEvenOrOdd()* // which accepts an integer value as argument.

Arguments: one argument of type integer

Return Type: an String value (Even/Odd/Invalid Input)

Specifications: The value returned by the method *isEvenOrOdd()* is determined by the following rules

if the given number is negative or zero, return "Invalid Input"

if the given number is even, return "Even"

if the given number is odd, return "Odd"

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_10_EvenOrOdd.java)***
2. **In the Template , add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_10_EvenOrOdd {
        public static void main(String[] args) {
        int num = 44;
        System.out.println(isEvenOrOdd(num));
        }

        public static String isEvenOrOdd(int num) {
        // ADD YOUR CODE HERE
        }
}
```

11. Rounder :

Define a method which returns the square of the given number if it is an even, return cube of the given number if it is an odd number.

Write the method with the following specifications

Name of method *calculate()* // which accepts an integer value as argument and return square of the given value if it is an even, return cube of the given value if it is an odd number.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *calculate()* is determined by the following rules

if the given number is negative or zero, return -1.

if the given number is even, return double the number.

if the given number is odd, return cube of the given number.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_11_Rounder.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_11_Rounder {

        public static void main(String[] args) {

        int num = 7;

        System.out.println(calculate(num));

        }

      public static int calculate(int num) {

        // ADD YOUR CODE HERE

        }


}
```

12. Sum Of Multiple 10

Define a method which returns the sum of three numbers after rounding off each number to the next multiple of 10. If any of the given number is multiple of 10 dont change it's value.

For Example:

if value of X is 56 round it's value to 60

if value of X is 30, don't change it's value.

Write the method with the following specifications

Name of method *sumOfMultiples()* // which accepts three integer value as argument and return the sum of three numbers after rounding off each number to the next multiple of 10.

Arguments: three argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *sumOfMultiples()* is determined by the following rules

if any of the given number is negative or zero, return -1.

in other case return the sum of all three rounded values.

Example

if a = 23, b = 34, c = 69

30 + 40 + 70 = 140

if a = 23, b = 34, c = 50

30 + 40 + 50 = 120

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_12_SumOfMul10.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_12_SumOfMul10 {

    public static void main(String[] args) {

    int a = 23, b = 34, c = 69;

    System.out.println(sumOfMultiples(a, b, c));

    }

    public static int sumOfMultiples(int a, int b, int c) {

    // ADD YOUR CODE HERE

    }

}
```

13. Rounded Sum:

Define a method which returns the sum of three rounded numbers. If the right most digit of the number is lessthan 5, then round off it's value to the previous multiple of 10 otherwise if the right most digit of the number is greater or equal to 5, then round off to the next multiple of 10.

Write the method with the following specifications

Name of method *sumOfRoundedValues()* // which accepts three integer value as argument and return the sum of three rounded numbers.

Example

if a = 23, b = 34, c = 66

20 + 30 + 70 = 120

if a = 23, b = 37, c = 55

20 + 40 + 60 = 120

Arguments: three argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *sumOfRoundedValues()* is determined by the following rules

if any of the given number is negative or zero, return -1.

if any of the given numbers right most digit is of the number is lessthan 5, then round off it's value to the previous multiple of 10 otherwise if the right most digit of the number is greater or equal to 5, then round off to the next multiple of 10.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_13_RoundedSum.java)***
2. **In the Template file, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_13_RoundedSum {
        public static void main(String[] args) {
        int a = 20, b = 30, c = 40;
        System.out.println(sumOfRoundedValues(a, b, c));
        }
        public static int sumOfRoundedValues(int n1, int n2, int n3) {
        // ADD YOUR CODE HERE
        }
}
```

14. AlaramClock:

Define a method which accepts two value as arguments(an integer and boolean) and return the string indicating when the alarm should ring.

the first argument indicating day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation or not.

Write the method with the following specifications

Name of method *ringAlarm()* // which accepts two arguments, first indicating day of the week and second a boolean indiacting if we are on vacation.

Arguments: two arguments of type integer and boolean

Return Type: an string value

Specifications: The value returned by the method *ringAlarm()* is determined by the following rules

if the first argument value is not between 0 to 6, return "Invalid Inputs"

if the second value is not boolean value true or false, return "Invalid Inputs"

if the first argument value is between 1 to 5 indicating the week day's and second value is true indicating on vacation, return "10:00"

if the first argument value is between 1 to 5 indicating the week day's and second value is false indicating not on vacation, return "07:00"

if the first argument value is 0 or 6 indicating the weekend day's and second value is true indicating on vacation, return "OFF"

if the first argument value is 0 or 6 indicating the weekend day's and second value is false indicating not on vacation, return "10:00"

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_14_AlarmClock.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_14_AlarmClock {
        public static void main(String[] args) {
        int day_of_week = 3;
        boolean onVac = true;
        System.out.println(ringAlarm(day_of_week, onVac));
        }
        public static String ringAlarm(int dayOfWeek, boolean onVac) {
        // ADD YOUR CODE HERE
        }
}
```

15. Boolean Value:

Define a method which accepts three boolean value as arguments and return true if any of the two values are true, other wise return false.

Write the method with the following specifications

Name of method *countBoolean()* // which accepts three boolean arguments, return true if any of the two values are true, else return false.

Arguments: two arguments of type boolean

Return Type: an boolean value

Specifications: The value returned by the method countBoolean() is determined by the following rules

if b1 = true, b2 = true, b3 = true then, return true

if b1 = true, b2 = true, b3 = false then, return true

if b1 = true, b2 = false, b3 = false then, return false

if b1 = false, b2 = false, b3 = false then, return false

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_15_BooleanValue.java)***
2. **In the Template file, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_15_BooleanValue {
        public static void main(String[] args) {
        boolean b1 = true, b2 = true, b3 = false;
        System.out.println(countBoolean(b1, b2, b3));
        }
        public static boolean countBoolean(boolean b1, boolean b2, boolean b3) {
        // ADD YOUR CODE HERE
   }
}
```

16. Natural Numbers:

Define a method which returns a String containing natural numbers between a range of two numbers separated by a single space. Solve it using While loops.

Write the method with the following specifications

Name of method : *getNaturalNumbers()*

Arguments : 2 Arguments of int type.

Return Type : A String value

Values must not be negative. If yes, then return -1 as string.

Values must not be 0. If yes, then return -2 as string.

Natural Numbers must be returned as one string with every value separated by single blankspace.

Consider that, the first argument value is less than the second argument number

Read the steps below carefully before you start

1. **User the skeleton code provided *(ECC_16_NaturalNumbers.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_16_NaturalNumbers
{

   public static void main(String[] arg) {
         int num1 = 10;
         int num2 = 20;
         System.out.println(getNaturalNumbers(num1, num2));
   }

   public static String getNaturalNumbers(int num1,int num2)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

17. Numbers In Range;

Define a method which returns a String contraining natural numbers between a range of two numbers separated by a single space.

Write the method with the following specifications

Name of method : *getNumbersInRange()*

Arguments : 2 Arguments of int type

Return Type : A String value

Values must not be negative. If yes, then return -1 as string.

Values must not be same. If yes, then return -2 as string.

1st arguments value must not be greater than 2nd arguments value. If yes, then return -3 as string.

Exclude the first and last value and return the result as one string with every value separated by single blankspace.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_17_NumbersInRange.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_17_NumbersInRange
{
   public static void main(String[] arg) {
         int num1 = 10;
         int num2 = 20;
         System.out.println(getNumbersInRange(num1, num2));
   }
   public static String getNumbersInRange(int num1,int num2)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

18. Reverse Order:

Define a method which returns a String contraining natural numbers between a range of two numbers separated by a single space.

Write the method with following specifications

Name of method : *getNumbersInRange()*

Arguments : 2 Arguments of int type

Return Type : A String value

Values must not be negative. If yes, then return -1 as string.

Values must not be same. If yes, then return -2 as string.

1st argument value must not be smaller than 2nd argument value. If yes, then return -3 as string.

Numbers in range must not include start value and end value.

Numbers in range must be returned as one string with every value separated by single blankspace.

Example: **input: 10 1**

output: 9 8 7 6 5 4 3 2

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_18_ReverseOrder.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_18_ReverseOrder
{
   public static void main(String[] arg) {
        int num1 = 20;
        int num2 = 39;
        System.out.println(getNumbersInRange(num1, num2));
   }
```

```
    public static String getNumbersInRange(int num1,int num2)
    {
            // ADD YOUR CODE HERE
            return null;
    }
}
```

19.Range With Step:

Define a method which returns a string of natural numbers within a range of two numbers with a given step/increment factor.

Write the method with following specifications

Name of method : *getNumbersInRange()*

Arguments : 3 arguments of type int

// first argument as start value

// second argument as end value

// third argument is the step/increment value

Return Type : A String value

Values must not be negative. If yes, then return -1 as string.

Values must not be same. If yes, then return -2 as string.

1st value must not be greater than 2nd value. If yes, then return -3 as string.

Numbers in range must not include start and end value.

Numbers in range must be returned as one string with every value separated by single blankspace.

For Example:

If input values are 10 30 2

Output: 11 13 15 17 19 21 23 25 27 29

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_19_RangeWithStep.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_19_RangeWithStep
{
   public static void main(String[] arg) {
        int num1 = 10;
        int num2 = 20;
        int num3 = 2;
        System.out.println(getNumbersInRange(num1, num2, num3));
   }
   public static String getNumbersInRange(int num1,int num2,int step)
   {
        // ADD YOUR CODE HERE
        return null;
   }
}
```

20.Four Per Line:

Define a method which returns a string of all numbers between 1 and the given input value.

Write the method with following specifications

Name of method : *getFourPerLine()*

Arguments : 1 argument of type int

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

Value must not be 0. If yes, then return -2 as string.

Value must not be greater than 99. If yes, then return -3 as string.

Numbers in range must be returned as one string with every value separated by single blankspace.

Ensure a new line after every set of 4 values.

For Example:

In Input: 12

Output:

1 2 3 4

5 6 7 8

9 10 11 12

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_20_FourPerLine.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_20_FourPerLine
{

   public static void main(String[] arg) {
         int num = 12;
         System.out.println(getFourPerLine(num));
   }
   public static String getFourPerLine(int num)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

21.BOX:

Define a method which accepts 2 numeric arguments and returns a box of the same size as a string with asterisk '*' symbol.

Write the method with following specifications

Name of method : *createBoxPattern()*

Arguments : 2 Integer Arguments // 1st argument represents Number of Rows.

// 2nd argument represents Numbe of Columns.

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

2) Value must not be 0. If yes, then return -2 as string.

1st value is rows and 2nd value is columns.

Box must be created using star symbol separated using a single blankspace for e.g. 1st value=4 & 2nd value=5 then output must be

```
* * * * *
*       *
*       *
* * * * *
```

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_21_Box.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

TEmplate:

```
public class ECC_21_Box
{
   public static void main(String[] args) {
         int rows = 2;
         int cols = 3;
         System.out.println(createBoxPattern(rows, cols));
   }
   public static String createBoxPattern(int rows,int cols)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

22. Star Pattern:

Define a method which accepts 1 numeric argument and returns a String of stars (*).

For instance if the given input is 3,then First line must have One star, Second Line Two stars, Third line Three Stars.

Write the method with following specifications

Name of method : *createStarPattern()*

Arguments : 1 Integer Argument // Represents Number of Rows.

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

Value must not be 0. If yes, then return -2 as string.

Pattern must be created using star symbol separated by single blankspace.

Example :

Input : 4

Output:

```
*
* *
* * *
* * * *
```

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_22_StarPattern1.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_22_StarPattern1
{

   public static void main(String[] arg) {
         int num = 30;
         System.out.println(createStarPattern(num));
   }

   public static String createStarPattern(int rows)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

23.Number Pattern:

Define a method which accepts 1 numeric argument and returns a pattern of numbers as a string.

Write the method with following specifications

Name of method : *NumberPattern4()*

Arguments : 1 Integer Argument // Represents Number of Rows.

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

Value must not be 0. If yes, then return -2 as string.

Value is rows.

Pattern must be created using numbers separated by single blankspace

Example :

Input: 5

Output:

1

2 4

3 6 9

4 8 12 16

5 10 15 20 25

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_23_NumberPattern4.java)***
2. **In the Template , add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

24. Check Prime:

Define a Method that checks for whether a given input is a prime number or not, and return a string.

Write the method with following specifications

Name of method : *checkPrime()*

Arguments : 1 Integer Argument

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

Value must not be 0 or 1. If yes, then return -2 as string.

If value is a prime number, then return true as string otherwise return false as string.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_24_CheckPrime.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_24_CheckPrime
{
   public static void main(String[] arg) {
         int num = 10;
         System.out.println(checkPrime(num));
   }
   public static String checkPrime(int num)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

25. Check Palindrom:

Define a Method that checks for whether a given input is a palindrome number or not, and return a string.

Write the method with following specifications

Name of method : *checkPalindrome()*

Arguments : 1 Integer Argument

Return Type : A String value

Value must not be negative. If yes, then return -1 as string.

Value must not be from 0 to 9. If yes, then return -2 as string.

If value is a palindrome, then return true as string otherwise return false as string.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_25_CheckPalindrome.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_25_CheckPalindrome
{


   public static void main(String[] arg) {
         int num = 121;
         System.out.println(checkPalindrome(num));
   }
\    public static String checkPalindrome(int num)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

26. CheckArmStrong :

Define a method which accepts a 4-digit value as number and checks whether the number is armstrong or not returns result.

Write the method with the following specifications

Name of method *checkArmStrong()* // which accepts an integer value as argument and return the String.

Arguments: one argument of type integer

Return Type: an String value

Value must not be negative. If yes, then return -1 as string.

Value must be a 4-digit number. If no, then return -2 as string.

If value is armstrong, then return "ArmStrong Number" as string otherwise return "Not ArmStrong Number" as string.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_26_CheckArmStrong.java)***
2. **In the Template , add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_26_CheckArmStrong
{
    public static void main(String[] arg) {
        int num = 153;
        System.out.println(checkArmStrong(num));
   }

   public static String checkArmStrong(int num)
   {
        // ADD YOUR CODE HERE
        return null;
   }
}
```

27. Factorial:

Define a method which which accepts a value as number and returns the factorial of the value.

Write the method with the following specifications

Name of method *getFactorial()* // which accepts an integer value as argument and return the factorial of the given value.

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *getFactorial()* is determined by the following rules

Value must not be negative. If yes, then return -1

Value must not be 0. If yes, then return -2

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_27_Factorial.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_27_Factorial
{
   public static void main(String[] arg) {
        int num = 3;
        System.out.println(getFactorial(num));
   }
   public static int getFactorial(int num)
   {
        // ADD YOUR CODE HERE
        return 0;
   }
}
```

28. List Factors:

Define a method which accepts a value as number and returns the factors of the value.

Write the method with the following specifications

Name of method *getFactors()* // which accepts an integer value as argument and return a String.

Arguments: one argument of type integer

Return Type: String value

Specifications: The value returned by the method getFactors() is determined by the following rules

1) Value must not be negative. If yes, then return -1 as string.

2) Value must not be 0. If yes, then return -2 as string

3) Factors must be returned as one string where every value is separated by a single blankspace.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_28_ListFactors.java)***
2. **In the Template add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_28_ListFactors
{

   public static void main(String[] arg) {
         int num = 20;
         System.out.println(getFactors(num));
   }

   public static String getFactors(int num)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

29. Sum Of Factors:

Define a method which accepts a value as number and returns the sum of factors of the value.

Write the method with the following specifications

Name of method *getSumOfFactors()* // which accepts an integer value as argument and return a String.

Arguments: one argument of type integer

Return Type: integer value

Specifications: The value returned by the method *getSumOfFactors()* is determined by the following rules

Value must not be negative. If yes, then return -1

Value must not be 0. If yes, then return -2.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_29_SumOfFactors.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_29_SumOfFactors
{

   public static void main(String[] arg) {
         int num = 20;
         System.out.println(getSumOfFactors(num));
   }
   public static int getSumOfFactors(int num)
   {
         // ADD YOUR CODE HERE
         return 0;
   }
}
```

30. Even Number Tester:

Define a method which accepts 2 values as number and returns the even numbers between the 2 values.

Note: If the first argument value is greater than second value, generate even numbers from first value to second value. If the second value is less than first value, generate even numbers from second value to first.

Write the method with the following specifications

Name of method *getEvenNumbers()* // which accepts 2 integer values as arguments and return the even numbers between the range.

Arguments: Two arguments of type integer

Return Type: String value

Specifications: The value returned by the method *getEvenNumbers()* is determined by the following rules

Values must not be negative. If yes, then return -1 as string.

Value must not be 0 or same. If yes, then return -2 as string.

The values must be returned as a single string where each value is separated by a single blankspace.

For Example:

Input: 10 20

Output: 10 12 14 16 18 20

Input: 20 10

Output: 10 12 14 16 18 20

Read the steps below carefully before you start

1. **Uer the skeleton code provided *(ECC_30_EvenNumberTester.java)***
2. **In the Template file, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_30_EvenNumberTester
{
   public static void main(String[] arg) {
         int num1 = 10;
         int num2 = 30;
         System.out.println(getEvenNumbers(num1, num2));
   }
   public static String getEvenNumbers(int num1,int num2)
   {

         // ADD YOUR CODE HERE
         return null;
   }
}
```

31. Find Square:

Define a method which returns the square of the given value

Write the method with the following specifications

Name of method *getSquare()*

Arguments: one argument of type integer

Return Type: an integer value

Specifications: The value returned by the method *getSquare()* is determined by the following rules

if the given number is 0, return -1

if the given number is negative value, return -2

for any positive value return square of the number

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_31_FindSquare.java)***
2. **In the Template file, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_31_FindSquare
{

   public static void main(String[] arg) {
         int num = 3;
         System.out.println(getSquare(num));
   }

   public static int getSquare(int num)
   {
         //ADD YOUR CODE HERE
         return 0;
   }
}
```

32.Find Triangle:

Define a method which accepts 3 values as number and checks and returns the name of the type of triangle generated.

Write the method with the following specifications

Name of method *findTriangle()*

Arguments: three arguments of type integers

Return Type: String value

Specifications: The value returned by the method *findTriangle()* is determined by the following rules

Values must not be 0. If yes, then return -1 as string.

Values must not be negative. If yes, then return -2 as string.

Sum of two sides must be greater than the third side. If no, then return -3 as string

If its a triangle with valid sides, then return as string whether the triangle formed is EQUILATERAL, ISOSCELES or SCALENE.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_32_FindTriangle.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_32_FindTriangle
{
   public static void main(String[] arg) {
         int num1 = 2;
         int num2 = 3;
         int num3 = 4;
         System.out.println(findTriangle(num1, num2, num3));
   }

   public static String findTriangle(int num1,int num2,int num3)
```

```
  {
        // ADD YOUR CODE HERE
        return null;
  }
}
```

33. ListPrimeNumbers:

Define a method which accepts 2 values as number and returns the prime numbers between the values.

Write the method with the following specifications

Name of method *getPrimeNumbers()*

Arguments: two arguments of type integers

Return Type: String value

Specifications: The value returned by the method *getPrimeNumbers()* is determined by the following rules

Value must not be negative. If yes, then return -1 as string.

1st value must not be greater than or equal to 2nd value. If yes, then return -2 as string.

Prime Numbers must be returned as one string with every value separated by single blankspace.

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_33_ListPrimes.java)***
2. **In the Template add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_33_ListPrimes
{

   public static void main(String[] arg) {
         int num1 = 20;
         int num2 = 30;
         System.out.println(getPrimeNumbers(num1, num2));
   }

   public static String getPrimeNumbers(int num1,int num2)
   {
         // ADD YOUR CODE HERE
         return null;
   }
}
```

34. PrimeNumbersSum:

Define a method which accepts two integer values as arguments and return the sum of prime numbers between the given range.

Write the method with the following specifications

Name of method *getPrimeNumbersSum()* // which accepts two integer values as argument and return the sum of all prime number between the range.

Arguments: two arguments of type integer

Return Type: an integer value

Specifications: The value returned by the method *getPrimeNumbersSum()* is determined by the following rules

Values must not be negative. If yes, then return -1

1st value must not be greater than or equal to 2nd value. If yes, then return -2

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_34_PrimeNumbersSum.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_34_PrimeNumbersSum
{

   public static void main(String[] arg) {
         int num1 = 10;
         int num2 = 20;
         System.out.println(getPrimeNumbersSum(num1, num2));
   }

   public static int getPrimeNumbersSum(int num1,int num2)
   {
         // ADD YOUR CODE HERE
         return 0;
   }
}
```

35.STringWeaver:

Define a method which which accepts 2 values as strings and returns a weaved string.

Write the method with the following specifications

Name of method *getWeavedString()*

Arguments: two arguments of type strings

Return Type: an string value

Specifications: The value returned by the method getWeavedString() is determined by the following rules

Values must not be blank. If yes, then return -1 as string.

If value1 is greater than value2 in length, then return a concatenated string which looks like value2+value1+value2.

If value1 is smaller than value2 in length, then return a concatenated string which looks like value1+value2+value1.

If value1 equal to value2 in length, then return a concatenated string which contains each character from both the values at the same position. For e.g."Hello" "Hello" result = "HHeelloo"

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_35_StringWeaver.java)***
2. **In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"**
3. **To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc**
4. **Compile your code**
5. **Check the output and upload the source file i.e., .java file**

Template:

```
public class ECC_35_StringWeaver
{

    public static void main(String[] arg) {
            String s1 = "Sudaksha";
            String s2 = "Education";
            System.out.println(getWeavedString(s1, s2));
    }

    public static String getWeavedString(String s1,String s2)
    {
            // ADD YOUR CODE HERE
            return null;
    }
}
```

36. CricketScore:

n Cricket-one day or T20, we often state the required score and the scoring rate. there are two ways to say it

a) 127 runs in 21.4 overs

b) 56 runs in 48 balls

The rule to decide which way to do it is as follows: if the number of runs required or the number of balls remaining is less than 100 use type (b), else use type (a).

A class called CricketScores is given to you. Implement a method called *getDisplayDetails(int runs, float Overs)*. This method should take the runs remaining and overs remaining.It should calculate the remaining overs and runs as per the logic given above and return the result String.

int runs: the required runs

float overs: a decimal number in the format overs.balls (example 21.4)

This function must return a string of the right type, combining with the run rate required. Thus the output will be either:

"XXX runs in ZZZ Overs @ YYY runs per over"

or

"XXX runs in WW balls @ YYY runs per ball"

NOTE: 21.4 overs implies 21 overs and 4 balls. The 0.4 is not to be taken as a regular fraction of an over. The XXX is an integer number, ZZZ is overs.balls figure, WW is integer number and YYY is a floating point number upto two decimals.

An example output would be

33 runs in 20 balls @ 1.5 runs per ball

130 runs in 20.4 Overs @ 6.12 runs per over

Read the steps below carefully before you start

1. **Use the skeleton code provided *(ECC_36CricketScores.java)***

2. In the Template, add your code in the placeholder - "ADD YOUR CODE HERE"
3. To write code, you can use editors such as Eclipse, Notepad, GEdit, VIM etc
4. Compile your code
5. Check the output and upload the source file i.e., .java file

Template:

```
public class ECC_36_CricketScores {


   public static String getDisplayDetails(int runs, float Overs){
         // ADD YOUR CODE HERE



         return null;
   }

   public static void main(String[] args) {
         // Test your code here


   }

}
```