Lab Assignment 04

User Input and Advanced Loops



CSE110: Programming Language I

| **No of Tasks** | | | **Points to Score** |
| --- | --- | --- | --- |
| **15** | | | **150** |

Submit the coding tasks (Task 1 - 11) on buX and the handwritten tasks (Task 12 - 15) to your Lab Instructors in the beginning of the next lab class.

1. Write a Java program that asks the user how many inputs they want to provide and then takes that many inputs and prints the maximum, minimum, and average of all the **even positive numbers** given by the user. If no even positive number is given, the average should be zero.

| **Sample Input** | **Sample Output** |
| --- | --- |
| 5  12  -8  19  8  -1 | Max: 12  Min: 8  Average: 10 |
| **Explanation:**  At first the user gave 5 as the input which indicates that the user will provide 5 numbers. Then 5 numbers were taken as inputs. Among these, only 12 and 8 are even positive numbers. | |

1. Write a Java program that will keep taking integer numbers as inputs from the user and print the square of those numbers until it gets a negative number and then stop.

**Sample Input/Output:** (The purple numbers are input.)

Enter Number: 2

2 ^ 2 = 4

Enter Number: 6

6 ^ 2 = 36

Enter Number: 1

1 ^ 2 = 1

Enter Number: 4

4 ^ 2 = 16

Enter Number: -5

1. Write a Java code that asks an integer as input from the user and takes that many integer inputs. Your task is to count how many numbers are non-negative and negative.

**Sample Input:** (The purple numbers are input.)

Enter an integer: 9

Enter number 1: -8

Enter number 2: 33

Enter number 3: -100

Enter number 4: 10

Enter number 5: 0

Enter number 6: 5

Enter number 7: 10

Enter number 8: -4

Enter number 9: 4

**Sample Output:**

6 Non-negative Numbers

3 Negative Numbers

1. Write a Java program to take a positive integer ***N*** (where N > 0) as user input and print the **first *N* prime numbers starting from 2**. Your code should check all the positive integers starting from 2 and determine whether they are prime or not until ***N*** prime numbers are found.

**Sample Input 1:**

5

**Sample Output 1:**

2

3

5

7

11

**Sample Input 2:**

7

**Sample Output 2:**

2

3

5

7

11

13

17

1. Write a Java code of a program that reads the value of N (where N > 0) from the user and calculates the value of y if the expression of y is as follows:

**Sample Input:**

The value of N: 2

**Sample Output:**

The value of y: -4

**Sample Input:**

The value of N: 4

**Sample Output:**

The value of y: -20

1. Write a Java program that will keep taking even positive integer numbers as inputs from the user and print the number of divisors(**factors**) of those numbers until it gets an odd number and then stops.

**Sample Input & Output:** (The purple numbers are input)

Enter Number: 44

44 has 6 divisors

Enter Number: 30

30 has 8 divisors

Enter Number: 8

8 has 4 divisors

Enter Number: 4

4 has 3 divisors

Enter Number: 6

6 has 4 divisors

Enter Number: 20

20 has 6 divisors

Enter Number: 24

24 has 8 divisors

Enter Number: 5

1. Read an integer N that is the number of test cases that follow. Each test case contains two integers X and Y. Print one output line for each test case that the sum of Y odd numbers from X including it if is the case. For example:

For the input 4 5, the output must be 45, that is: 5 + 7 + 9 + 11 + 13

For the input 7 4, the output must be 40, that is: 7 + 9 + 11 + 13

| **Sample Input** | **Sample Output** |
| --- | --- |
| 2  4  3  11  2 | 21  24 |
| Explanation: Here, the 2 means there are two test cases. For each test case you have to take two inputs (X, Y) and print the sum of Y odd numbers starting from X. | |

1. Take the length and width of a **rectangle** from the user and create the rectangle according to the output below. Your output should match the specified output.

| **Sample Input #1**  4  6 | **Sample Input #2**  3  5 |
| --- | --- |
| **Output**  1 2 3 4  1 2 3 4  1 2 3 4  1 2 3 4  1 2 3 4  1 2 3 4 | **Output**  1 2 3  1 2 3  1 2 3  1 2 3  1 2 3 |

1. Take the height of a **right-justified right triangle** from the user and create the triangle according to the output below. Your output should match the specified output.

| **Sample Input #1**  4 | **Sample Input #2**  3 |
| --- | --- |
| **Output**  1  1 2  1 2 3  1 2 3 4 | **Output**  1  1 2  1 2 3 |

1. Take the height of an **isosceles triangle** from the user and create the triangle according to the output below. Your output should match the specified output.

| **Sample Input #1**  4 | **Sample Input #2**  3 |
| --- | --- |
| **Output**  1  1 2 3  1 2 3 4 5  1 2 3 4 5 6 7 | **Output**  1  1 2 3  1 2 3 4 5 |

1. Write a Java program that will ask for a range (a starting number and an ending number) from the user and print all the Armstrong numbers between that range.

***[Armstrong Number:****An Armstrong number is a number whose sum of digits raised to the power the number of digits equals to that number.*

*For example, 371 is an Armstrong number because 33 + 73 + 13 = 371, here the total number of digits in 371 is 3* ***]***

**Sample Input 1:**

Start: 300

End: 500

**Sample Output 1:**

Armstrong numbers:

370

371

407

**Sample Input 2:**

Start: 100

End: 200

**Sample Output 2:**

Armstrong numbers:

153

# 

1. Trace the following code, create a tracing table and write the outputs.

| **1** | **public class T1{** |
| --- | --- |
| **2** | **public static void main(String args[]){** |
| **3** | **int x = 0, y = 0;** |
| **4** | **int sum = 0;** |
| **5** | **while (x < 4){** |
| **6** | **y = x - 3;** |
| **7** | **while (y < 3){** |
| **8** | **sum = (sum % 3) + x - y \* 3 ;** |
| **9** | **System.out.println(sum);** |
| **10** | **y = y + 1;** |
| **11** | **}** |
| **12** | **if (x > 5){** |
| **13** | **x++;** |
| **14** | **}** |
| **15** | **else{** |
| **16** | **x += 2;** |
| **17** | **}** |
| **18** | **}** |
| **19** | **}** |
| **20** | **}** |

1. Trace the following code, create a tracing table and write the outputs.

| **1** | **public class T2 {** |
| --- | --- |
| **2** | **public static void main(String args[]) {** |
| **3** | **int x = 0, i = 0, sum = 0;** |
| **4** | **i = 1;** |
| **5** | **x = 2;** |
| **6** | **sum = 0;** |
| **7** | **while (i < 20){** |
| **8** | **x = x + i;** |
| **9** | **sum = sum + x + 1;** |
| **10** | **System.out.println(sum);** |
| **11** | **if (x > 5){** |
| **12** | **i += 2;** |
| **13** | **}** |
| **14** | **else {** |
| **15** | **i += 3;** |
| **16** | **}** |
| **17** | **}** |
| **18** | **sum = sum + i;** |
| **19** | **System.out.println(sum);** |
| **20** | **}** |
| **21** | **}** |

1. Trace the following code, create a tracing table and write the outputs.

| **1** | **public class T3** |
| --- | --- |
| **2** | **{** |
| **3** | **public static void main(String args[])** |
| **4** | **{** |
| **5** | **int x = 0, y = 0;** |
| **6** | **int sum = 0;** |
| **7** | **while (x < 10){** |
| **8** | **y = x - 3;** |
| **9** | **y = 40;** |
| **10** | **while (y > 22){** |
| **11** | **if ((sum > 30) && (sum < 40)){** |
| **12** | **sum = sum + x \* 2 ;** |
| **13** | **}** |
| **14** | **else if ((sum > 40) && (sum < 50)){** |
| **15** | **sum = sum + x \* 3;** |
| **16** | **}** |
| **17** | **else {** |
| **18** | **sum = sum + 23;** |
| **19** | **}** |
| **20** | **System.out.println(sum);** |
| **21** | **y = y - 10;** |
| **22** | **}** |
| **23** | **x += 2;** |
| **24** | **}** |
| **25** | **}** |
| **26** | **}** |

1. Trace the following code, create a tracing table and write the outputs.

| **1** | **public class T4{** |
| --- | --- |
| **2** | **public static void main(String args[]){** |
| **3** | **boolean check = true;** |
| **4** | **int x = 2, y = 2, z = 3;** |
| **5** | **while(check){** |
| **6** | **y = 4 / x % 3 + z \* y - 5;** |
| **7** | **if(y > 10 || x==7){** |
| **8** | **z += 3;** |
| **9** | **break;** |
| **10** | **}** |
| **11** | **if(4+x%3 > 5){** |
| **12** | **x %= y + (z--) + z;** |
| **13** | **System.out.println(x);** |
| **14** | **}** |
| **15** | **else{** |
| **16** | **y += x + (--z) + y;** |
| **17** | **System.out.println(y);** |
| **18** | **}** |
| **19** | **x++;** |
| **20** | **System.out.println(x + y);** |
| **21** | **}** |
| **22** | **}** |
| **23** | **}** |