
Lab Assignment 3

Linear Loops



CSE110: Programming Language I

	Lab Tasks	Home Tasks
Coding	7	9
Tracing	1	2

[You need to submit **only the Home Tasks**. Submit all the Homework Coding Tasks (Task 1 to 9) in the Google Form shared on buX. Submit all the Homework Tracing Tasks (Task 10 & 11) handwritten to your Lab Instructors before the next lab.]

LAB Tasks

(No Need to Submit)

1. Write a Java code that would print the following sequences **using while/for loop**:
 - a) 24, 18, 12, 6, 0, -6
 - b) 18, -27, 36, -45, 54, -63
2. Write a Java program that will take N numbers from the user and find their sum and average **using a for loop**.

Sample Input:

N = 5

Input the 5 numbers:

1
2
3
4
5

Expected Output:

The sum of 5 no is: 15

The Average is: 3.0

3. Write a Java program which adds all natural numbers that are multiples of both 7 and 9 up to 600. *[Ans: 2835]*
4. Write a Java program which adds all numbers that are multiples of either 7 or 9 or both up to 600. Ensure that numbers like 63 are added only once in the sum. *[Ans: 42649]*
5. 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

6. Write a Java program that will calculate the value of y where the expression of y is as follows :

$$y = 1^2 - 2^2 + 3^2 - 4^2 + 5^2 \dots\dots\dots + n^2$$

Sample Input	Output	Explanation
5	15	$y = 1 - 4 + 9 - 16 + 25 = 15$
10	-55	$y = 1 - 4 + 9 - 16 + 25 - 36 + 49 - 64 + 81 - 100 = -55$
20	-210	$y = 1 - 4 + 9 - 16 + 25 \dots\dots\dots - 400 = -210$

7. Write a program in Java that asks the user for an **integer** input and counts the number of digits in the number.

Hint: You may keep dividing the number by ten and count how many times this can be done until the number becomes 0.

Sample Input	Output
7546	Total digits = 4

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

1	<code>public class Trace1{</code>
2	<code> public static void main(String args[]){</code>
3	<code> int x = 0, p = 0, sum = 0;</code>
4	<code> p = 1;</code>
5	<code> x = 2;</code>
6	<code> double q;</code>
7	<code> sum = 0;</code>
8	<code> while (p < 12){</code>
9	<code> q = x + p-(sum+7/3)/3.0%2 ;</code>
10	<code> sum = sum + (++x) + (int)q;</code>
11	<code> System.out.println(sum++);</code>
12	<code> if (x > 5){</code>
13	<code> p += 4/2;</code>
14	<code> }</code>
15	<code> else {</code>
16	<code> p += 3%1;</code>
17	<code> }</code>
18	<code> }</code>
19	<code> sum = sum + p;</code>
20	<code> System.out.println(sum);</code>
21	<code> }</code>
22	<code>}</code>

Home Tasks

1. Write a Java program that will take a positive integer n as input and print all the numbers from 0 to n which are **divisible by 5 but not divisible by 3**.

Sample Input

40

Sample Output

5
10
20
25
35
40

2. Write a Java program which adds all numbers that are multiples of either 7 or 9 but not both, up to 600. Use only a single loop in your program. *[Ans: 39814]*
3. 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

4. Write a Java program that displays the sum of first n odd natural numbers.

Sample Input:

Input number of terms: 5

Expected Output:

The odd numbers are:

1

3

5

7

9

The Sum of odd Natural Numbers up to 5 terms is: 25

5. Write a Java program that will read 10 numbers from the user, and then print the first number, the sum of the first 2 numbers, the first 3 numbers, and so on up to the sum of 10 numbers.

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

Enter Number: 6

Sum = 6

Enter Number: 7

Sum = 13

Enter Number: 2

Sum = 15

Enter Number: 8

Sum = 23

Enter Number: 7

Sum = 30

Enter Number: 5

Sum = 35

Enter Number: 1

Sum = 36

Enter Number: -12

Sum = 24

Enter Number: 0

Sum = 24

Enter Number: 1

Sum = 25

6. Write a Java program that will take an integer as input and print all the divisors of that number.

Sample Input

6

Sample Output

Divisors of 6:

1

2

3

6

7. Write a program in Java that prints the individual digits of an integer number (user input) backward (From right to left).

Sample Input	Output
32768	8, 6, 7, 2, 3

Hint: First to get the digit from the right side, we can take the remainder of the number using the modulus (%) operator i.e. mod 10 to get the rightmost digit and print it. For dropping the last digit, we can perform division by 10 on the number and then continue the same to print the other digits as shown below.

$$32768 \% 10 = 8$$

$$32768 / 10 = 3276$$

Then,

$$3276 \% 10 = 6$$

$$3276 / 10 = 327$$

and so on

$$327 \% 10 = 7$$

$$327 / 10 = 32$$

$$32 \% 10 = 2$$

$$32 / 10 = 3$$

$$3 \% 10 = 3$$

$$3 / 10 = 0$$

Done! When the number becomes 0 you can end your loop.

8. Write a program in Java that asks the user for an integer input, and print the individual digits forward (From left to right). *[You are not allowed to use Math.pow() to solve this problem.]*

Sample Input	Output
32768	3, 2, 7, 6, 8

[Hint: First, count how many digits. Then calculate 10 to the power that (number of digits) minus 1. Say, 32768 has 5 digits, so you calculate 10 to the power 4 which is 10,000.

Then divide 32,768 by 10,000 and thus you get 3

Take remainder of 32,768 by 10,000 and thus you get 2,768

Then divide 10,000 by 10 to get 1,000

Then divide 2,768 by 1,000 and thus you get 2.

take remainder of 2,768 by 1,000 and thus you get 768

keep going on until there are no more digits left (zero!).

In short:

Part 1: First count digits, say 5 in this case for 32,768

Part 2: Then calculate 10 to the power 4 (5-1), that is 10,000.

Part 3: Then repeat the following three steps:

$$32,768 / 10,000 = 3$$

$$32,768 \% 10,000 = 2,768$$

$$10,000 / 10 = 1,000$$

$$2,768 / 1,000 = 2$$

$$2,768 \% 1,000 = 768$$

$$1,000 / 10 = 100$$

$$768 / 100 = 7$$

$$768 \% 100 = 68$$

$$100 / 10 = 10$$

$$68 / 10 = 6$$

$$68 \% 10 = 8$$

$$10 / 10 = 1$$

$$8 / 1 = 8$$

$$8 \% 1 = 0$$

$$1 / 10 = 0$$

9. Write a Java program that will take an integer as input and -

- a) Find out if the number is a prime number or not.
- b) Find out if the number is a perfect number or not.

[Prime Number: If a number has **only two divisors**, (1 and itself), then it is a prime number. Else, then it is not a prime number.

Perfect Number: A number is said to be a perfect number if the **sum of its divisors**, including 1 but not the number itself is equal to that number.]

Sample Input

6

Sample Output

6 is not a prime number

6 is a perfect number

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

1	<code>public class Trace2 {</code>
2	<code> public static void main(String[] args) {</code>
3	<code> int x = 0, p = 0, sum = 0;</code>
4	<code> p = 1;</code>
5	<code> x = 2;</code>
6	<code> double q = 0.0;</code>
7	<code> sum = 5;</code>
8	<code> while (p < 15) {</code>
9	<code> q = x + p - (sum + (int) (7 / 4)) / 3.0 % 2;</code>
10	<code> sum = sum + x + (int) q;</code>
11	<code> x += 1;</code>
12	<code> System.out.println(sum);</code>
13	<code> if (x > 5) {</code>
14	<code> p += (int) (5 / 2);</code>
15	<code> }</code>
16	<code> else {</code>
17	<code> p += 10 % 3;</code>
18	<code> }</code>
19	<code> }</code>
20	<code> sum = sum + p;</code>
21	<code> System.out.println(sum);</code>
22	<code> }</code>
23	<code>}</code>

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

1	<code>public class Trace3 {</code>
2	<code> public static void main(String[] args) {</code>
3	<code> int m = 17, n = 13, p = 1, sum = 30;</code>
4	<code> while(0 < p%10){</code>
5	<code> if(m % 10 == 0){</code>
6	<code> sum = sum * m % n + p / n ;</code>
7	<code> }</code>
8	<code> else{</code>
9	<code> if(m % 4 == 0){</code>
10	<code> sum += n + (--m) ;</code>
11	<code> }</code>
12	<code> else{</code>
13	<code> sum -= m--;</code>
14	<code> }</code>
15	<code> }</code>
16	<code> p+=1;</code>
17	<code> System.out.println(sum) ;</code>
18	<code> }</code>
19	<code> System.out.println(!(n%13 == 0) && !false p>10);</code>
20	<code> }</code>
21	<code>}</code>