Task 1

Write the C programs, which prints the following sequences of values in loops:

a) 24, 18, 12, 6, 0, -6

b) -10, -5, 0, 5, 10, 15, 20

c) 18, 27, 36, 45, 54, 63

d) 18, -27, 36, -45, 54, -63

==========

Hint (1): You may use a while loop for solving these problems.

Hint (2): We are already familiar with the print() function. But, when we use it to print any value, it automatically adds an additional newline after each print statement.

For example:
print(1) print(2)

Output:
1
2

==========

To solve this problem, in C, we can add an extra argument (end = " ") in the print function which tells the program to skip printing the additional newline.

For example: print(1,
end ="") print(2)

Output: (prints the next output right to the previous one)
12

==========

In Task-1(a), the loop counter may be initialized at 24 and then the loop should terminate when the loop counter reaches -6. The difference between the first two values is 24 - 18 = 6. So, the loop counter value is getting decremented by 6 in every iteration for the given sequence here.

For your understanding, the code for Task 1(a) is done for you.

```
# a) 24, 18, 12, 6, 0, -6

# initializing loop counter counter
= 24

# loop structure while
counter >= -6:

    #inside loop body
if counter == -6:
        print(counter, end = "")
else:
        print(counter, end = ", ")
          counter = counter - 6 #updating loop
counter
    #inside loop body

#outside loop body
```

Hints for 1(d):

print(5 * (-1)) gives output -5
print("-" + str(5)) gives output -5

Task 2

Write a C code for the following:

1) Ask the user to enter the name of his favorite car. 2)
Ask the user to enter a Number

Display the name of the user's favorite car, the number of times specified in the second step.

==========

Example 1: If the user enters "Toyota" and 2, your program should print the name "Toyota" two times.

Sample Input 1:
Toyota 2

Sample Output 1:
Toyota
Toyota

==========

Example 2: If the user enters “Veyron” and 5, your program should print the name “Veyron” five times.

Sample Input 2:
Veyron 5

Sample Output 2:
Veyron
Veyron
Veyron
Veyron
Veyron

Task 3

Write the C code of a program that adds all numbers that are multiples of **both 7 and 9** up to 600 (including 600) i.e. 63, 126, 189, 252, ...

The output of your program should be: 2835
since 63 + 126 + 189 + 252 + 315 + 378 + 441 + 504 + 567 = 2835

Task 4

Write a C code of a program that adds all numbers that are multiples of **either 7 or 9 but not both**, up to 600 (including 600) i.e. 7, 9, 14, 18, 21..... and so on but not the numbers 63, 126, 189..... which are multiples of both 7 and 9.

The output of your program should be: 39814

Task 5

Write the C code of a program that displays all the **odd numbers** between 10 and 50 (inclusive).

Output: 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49

Task 6

Write a C code that will calculate the **value of y if the expression** of y is as follows (n is the input): $y=1^2-2^2+3^2-4^2+5^2.....+n^2$

==========

Sample Input 1: 5

Sample Output 1:
15

Explanation: y = 1 - 4 + 9 - 16 + 25 = 15

==========

Sample Input 2:
10

Sample Output 2:
-55

Explanation: y = 1 - 4 + 9 - 16 + 25 - 36 + 49 - 64 + 81 - 100 = -55

==========

Sample Input 3:
20

Sample Output 3:
-210

Task 7

Write a C code of a program that asks the user to enter ten numbers and then display the total and the average of **ONLY** the **odd numbers** among those ten numbers.
[Please do not use list for this task]

==========

Sample Input 1:
1

2

3

4

5

6

7

8

9

10

Sample Output 1: The total of the odd numbers is 25 and their average is 5.0

Explanation: The total is 1 + 3 + 5 + 7 + 9 = 25 and the average is 25/5 = 5.0

==========

Sample Input 2:
-20

13

-5

40

-17

10

20

-8

99

-200

Sample Output 2:

The total of the odd numbers is 90 and their average is 22.5

Explanation: The total is 90 = 13 + (-5) + (-17) + 99) and their average is 90/4 = 22.5

Task 8

Write a C code for the following:

- Ask the user to enter a Number, N
- Display the summation of multiples of 7 up to that number (**from 1 to N inclusive**)

==========

Sample Input 1:
50

Sample Output 1:
196

Explanation: The summation of multiples of 7 up to 50 is 7 + 14 + 21 + 28 + 35 + 42 + 49 = 196

==========

Sample Input 2:
75

Sample Output 2:
385

Explanation: The summation of multiples of 7 up to 75 is 7 + 14 + 21 + 28 + 35 + 42 + 49 + 56 + 63 + 70 = 385

Task 9

Write a C code that will read 5 numbers from the user. Your program should print the first number, the sum of the first 2 numbers, the sum of the first 3 numbers, and so on up to the sum of 5 numbers.

==========

Sample Input 1:

1
2
3
4
5

Sample Output 1:

1
3
6
10
15

Explanation:

When the user enters 1 at first, we print 1, then
The user enters 2, (1 + 2) = 3, so our output is 3, then
The user enters 3, (3 + 3) = 6, hence our output is 6, then
The user enters 4, and we have (6 + 4) = 10, we print 10, and finally The
user enters 5, the final sum is 15 which is printed as the last output.

==========

Sample Input 2:

11
-29
-3
47
50

Sample Output 2:

11
-18
-21
26
76

Explanation:

When the user enters 11 at first, we print 11, then
The user enters -29, (11 + (-29)) = -18, so our output is -18, then
The user enters -3, (-18 - 3) = -21, hence our output is -21, then
The user enters 4, and we have (-21 + 47) = 26, we print 26, and finally
The user enters 50, the final sum is (26 + 50) = 76 which is printed as the last output.

Task 10

Write a C program which takes a number and prints the digits from the unit place, then the tenth, then hundredth, etc. (Right to Left)

[Consider the input number to be an INTEGER. You are not allowed to use String indexing for solving this task]

Example: If the user gives 32768, then print 8, 6, 7, 2, 3

==========

Hint (1): The input() function, converts the input data to String data type by default. Therefore, please type cast it to an integer before proceeding further.

Hint (2): First to get the digit from the right side, we can take the remainder of the number using modulus (%) operator i.e. mod 10 to get the rightmost digit and print it. For dropping the last digit, we can perform floor 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.

Task 11

Write a C program that takes a number and prints how many digits are in that number.
[Consider the input number to be an INTEGER.]

[You are not allowed to use len() function]

Example: If the user gives 9876, your program should print 4.

Hint: We may keep floor dividing by ten and count how many times this can be done until the number becomes 0 as illustrated below.

9876 // 10 = 987, count = 1 then,
987 // 10 = 98, count = 2
98 // 10 = 9, count = 3 9
// 10 = 0, count = 4

Done! When the number becomes 0 your loop can end giving you the count of digits, in this case for our input of 9876, 4 digits.

Task 12

Write a C program that takes a number from the user and prints its digits from left to right.

[Consider the input number to be an INTEGER. You are not allowed to use String indexing for solving this task]

Example: if the user gives 32768, then print 3, 2, 7, 6, 8

==========

Hint(1): The input() function takes the input data as String data type by default. Please convert it to an integer before starting your code for the task.

Hint(2):

Step 1: First, count how many digits
Step 2: Then, calculate 10 to the power (number of digits - 1).

Step 3 with explanation: Say, the input given by the user as in our example, 32768 has 5 digits, so calculating 10 to the power 4 gives us 10000. Then floor dividing 32768 by 10000, we can get 3.

Proceeding further, the remainder of 32768 by 10000 (32768 % 10000) gives us 2768. This time to get 2 we need to floor divide 2768 by 1000 which is basically our 10000/10. Again, taking remainder of 2768 by 1000 gives us 768 which we then divide by 100 (i.e. 1000/10) and keep on doing this until there are no more digits left (zero).

To summarize and clarify:
Loop 1: First, we count digits like our Task 12, say 5 in this case for 32768 Loop
2: Then, we calculate 10 to the power 4 (5-1), that is 10000.
Loop 3: Then we keep repeating the three steps of floor dividing, modulus and dividing by 10 as demonstrated below.

32768 // 10000 = 3
32768 % 10000 = 2768
10000 // 10 = 1000

2768 // 1000 = 2
2768 % 1000 = 768 1000
// 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

Done. Loop ends as number has become 0.

13

Write a C program that takes a number as input from the user and prints the divisors of that number as well as how many divisors the number has.

==========

Sample Input 1: 6

Sample Output 1:
1, 2, 3, 6
Total 4 divisors.

==========

Sample Input 2:
121

Sample Output 2:
1, 11, 121
Total 3 divisors.

Task 14

Write a C program that takes a number as input from the user and tells if it is a perfect number or not.

Perfect Number: An integer number is said to be a perfect number if its factors, including 1 but not the number itself, sum to the number.

==========

Sample Input 1: 6

Sample Output 2: 6
is a perfect number

Explanation:
6 has 4 divisors: 1, 2, 3, and 6.
If we add all factors except 6 itself, 1 + 2 + 3 = 6. The sum of the factors excluding the number itself sum up to the number therefore "6 is a perfect number" is printed.

==========

Sample Input 2:
28

Sample Output 2:
28 is a perfect number

Explanation:

Task

28 has the divisors: 1, 2, 4, 7, 14, and 28.
If we add all factors except itself, we get 28, 1 + 2 + 4 + 7 + 14 = 28.

==========

Sample Input 3:
33

Sample Output 3: 33 is
not a perfect number

Explanation:
33 has 4 divisors: 1, 3, 11, and 33.
If we add all factors except itself, 15 = 1 + 3 + 11. The sum is not equal to the number, therefore 33 is not a perfect number.

Task 15

Write a C program that asks the user for one number and tells if it is a prime number or not.

Prime Number: If a number has only two divisors, (1 and itself), then it is a prime number. If it is divisible by more numbers, then it is not a prime.

Hint: You may use the code for counting divisors from Task 14.

==========

Sample Input 1:
11

Sample Output 1:
11 is a prime number

Explanation: 11 has only 2 divisors: 1 and 11.

==========

Sample Input 2: 6

Sample Output 2:
6 is not a prime number

Explanation: 6 has 4 divisors: 1, 2, 3 and 6.

16

Write a C program that asks the user for a quantity, then takes that many numbers as input and prints the maximum, minimum and average of those numbers.

[Please note that you CANNOT use max, min built-in functions]

[Also, you DO NOT need to use lists for this task]

==========

Example: If the user enters 5 as an input for quantity and the enters the 5 numbers, 10, 4, -1, 100, and 1. The output of your program should be: “Maximum 10”, “Minimum -100”, “Average is -17.2” as shown below.

Input: 5 10 4 -1
-100 1

Output:
Maximum 10
Minimum -100
Average is -17.2

Explanation: Average calculation: (10 + 4 + (-1) + (-100) + 1)/5 = -86/5 = -17.2

Task 17

Write a C program that prints a square of size N using + where N will be given as input as illustrated in the examples below.

Hint: You may need to use nested loops to solve this problem.

==========

Sample Input 1:

5

Sample Output 1:

```
+++++
+++++
+++++
+++++
+++++
```

Explanation: Here, we may see it as 5 rows and 5 columns, where in each horizontal row/line, 5 '+' is printed next to each other and we have a total of 5 lines.

==========

Sample Input 2:

3

Sample Output 2:

```
+++
+++
+++
```

Task

Explanation: If the input is 3, then there will be 3 rows and 3 columns, where in each horizontal row/line, '+' is printed 3 times, 1 in each of the 3 columns and we will have a total of 3 lines.

Task 18

Write a C program that prints a rectangle of size M (height/line numbers) and N (length/column numbers) using incrementing numbers where M and N will be given as input. Please look at the examples below for clarification.

Hint: You may need to use nested loops and print the loop counter variable in one of the loops.

==========

Sample Input 1:

4
6

Sample Output 1:

123456
123456
123456
123456

Explanation: The user has given 4 rows/lines and 6 columns as input. Therefore, we have 4 lines in our output here and in each line, the column number is printed from 1 through to 6 for the 6 columns.

==========

Sample Input 2:

3
2

Sample Output 2:

12
12
12

Explanation: Our user input this time is 3 rows/lines and 2 columns. So, our output has 3 lines and, in each line, the column number 1 and 2 are printed next to each other as the user only wants 2 columns here.

Write a C program that prints a right-angled triangle of height N using incrementing numbers where N will be given as input.

Hint: You may need to use nested loops. Try to think up to which point should the inner loop run.

==========

Sample Input 1: 4

Sample Output 1:

1
12
123
1234

Explanation: For an input of 4, we have 4 rows/lines where in each line, the respective column number is printed sequentially up to the line/row number. So, in line number 1, we have 1 only. In line 2, 12 is printed. In line 3, we have 123 and so on.

==========

Sample Input 2: 5

Sample Output 2:

1
12
123
1234
12345

Explanation: Numbers are printed sequentially up to the line number for each of the lines.