CSE110 Homework 4

Task 1:

Write **javacode** for the following:

1) Ask the user to enter the name of his favorite car.
2) Ask the user to enter a Number
3) Display the name of the user's favorite car, number of time specified in the second step.

Example: If the user enters "Toyota" and 20, your program should print the name Toyota twenty times.

Task 2:

Write **javacode** to display all the odd numbers between 10 and 50.

Task 3:

Write **javacode** for the following:

Take twenty numbers input from the user and print the maximum and the average.

Task 4:

Write **javacode** for the following:

Take twenty numbers input from the user and find the minimum from all numbers and the average of the even numbers entered by the user. [If the user enters odd numbers ignore them]

Task 5:

Write **javacode** that reads the value of *n* and calculates the value of y if the expression of y is as follows:

$$y^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 + 6^3 + \ldots\ldots\ldots\ldots + n^3$$

Task 6:

Write **javacode** that will calculate the value of y if the expression of y is as follows (n is the input):

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

Task 7:

Write **javacode** of a program which adds all numbers that are multiples of both 7 and 9 up to 600.

Task 8:

Write **javacode** of a program which adds all numbers that are multiples of either 7 or 9 up to 600. Ensure that numbers like 63 are added only once in the sum.

Task 9:

Write **javacode** of a program which adds all numbers that are multiples of either 7 or 9 but not both, up to 600.

Task 10:

Write **javacode** of a program that asks the user to enter ten numbers then display ONLY the total and the average of the odd numbers among those ten numbers.

[Hint: Example Input: 1 2 3 4 5 6 7 8 9 10 and Example Output: Total is 25 and Average is 5 (i.e., Total is 25 = (1+3+5+7+9) and Average is 25/5 = 5)]

Task 11:
Solve Task 10 for even numbers instead of odd numbers.
Task 12:
Solve Task 10 for numbers that are multiples of 4, instead of odd numbers.

Task 13:
Write **javacode** of a program that reads a number N, and prints out the sum of all odd numbers from 1 to N inclusive. For instance, if the input is 6, the output for the program should be 9.

Task 14:
Write **javacode** of a program that reads a list of numbers, and prints out the product of all the numbers read. You may assume that the user first inputs the total number of numbers. For example, if the first input is 4, then the program has to read in four numbers from the user, and print out the product of these four numbers. Assume that user will never enter first number as zero.

Task 15:
Write **javacode** of a program that will read 20 numbers from the user, and then print the sum of first number, then sum of the first 2 numbers, sum of first 3 numbers, and so on up to the sum of 20 numbers.

Task 16:
Write **javacode** of a program that reads marks of ten courses and prints the maximum, minimum and average of those ten marks.

Task 17:
In mathematics, the Fibonacci numbers form a sequence defined by the following recurrence relation:

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

That is, after two starting values, each number is the sum of the two preceding numbers. The first Fibonacci numbers, also denoted as Fn, for n = 0, 1, … , are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,

Write **javacode** of a program, which prints all Fibonacci numbers that are less than 1600.

Task 18:
Write **javacode** of a program which takes a number and tells how many digits are in that number.
Example: if user gives 9876, you should print 4.
Hint: keep dividing by ten and count how many times this could be divided.
9876 by 10, is 987, count that got 1 digit
987 by 10, is 98, count that got 1 digit (total 2)
98 by 10, is 9, count that got 1 digit (total 3)
9 by 10, is 0, count that got 1 digit (total 4)
done!

Task 19:

Write **javacode** of a program which takes a number and prints the value of 10 to the power that number. You need to use loop because variable in the power isn't allowed.
For example: if user gives 3, print 1000.
Hint: Keep multiplying 1 by 10, again and again, 3 (or n) times
like sum=sum+n, you need to write, product = product x 10
1 x 10 = 10
10 x 10 = 100
100 x 10 = 1000

Task 20:

Write **javacode** of a program which takes a number and prints the digits from unit place, then tenth, then hundredth, etc.
Example: if user gives 32768, then print 8, 6, 7, 2, 3
Hint: Taking remainder/modulus of division by 10.
After printing the remainder, drop the last digit by dividing by 10. Then start over.

32,768 % 10 = 8
32,768 / 10 = 3, 276

3, 276 % 10 = 6
3, 276 / 10 = 327

327 % 10 = 7
327 / 10 = 32

32 % 10 = 2
32 / 10 = 3

3 % 10 = 3
3 / 10 = 0

Task 21:

Write **javacode** of a program which takes a number and prints the digits from left to right.
Example: if user gives 32768, then print 3, 2, 7, 6, 8
Hint: count how many digits
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 is no more digits left (zero!).

In short:
Loop 1: First count digits, say 5 in this case for 32,768
Loop 2: Then calculate 10 to the power 4 (5-1), that is 10,000.
Loop 3: Then repeat 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

Task 22:
Telling words: Write **javacode** of a program that takes a number between 0 and 9. You have to print that number in words.
Hint: if n == 1, then print "one"
else if n == 2, then print "two"

Task 23:
Combine Task 21 and 22 into a single **program** so that it can tell any number in words.
Example: If user gives 932, print nine three two.

Task 24:
Write **javacode** of a program that takes a number and prints all numbers up to that number.
If the user gives 8, print 1 to 8.

Task 25:
Write **javacode** of a program that takes a number and counts how many times that number can be divided by all numbers up to that number (Those numbers are also known as factors)
If the user gives 8, tries to divide 8 by each of 1 to 8 and count how many times it could be divided.

For example:
If user enters 8,
try to divide 8 by 1, its divisible (increase count to 1)
try to divide 8 by 2, its divisible (increase count to 2)
try to divide 8 by 3, its NOT divisible
try to divide 8 by 4, its divisible (increase count to 3)
try to divide 8 by 5, its NOT divisible
try to divide 8 by 6, its NOT divisible
try to divide 8 by 7, its NOT divisible
try to divide 8 by 8, its divisible (increase count to 4)
Now print the count which is 4 in this case.

Task 26:
If a number is NOT divisible any number other than 1 and itself, then it is called prime number.
For example, 13 is a prime number because it is NOT divisible by any number other than 1 and 13 (itself).
Take one number from the user and tell if it is prime number or not.
Hint: Use the technique from Task 25 and count factors of the input. Factors are those numbers between 1 and n that can divide the number, n. If there are more than two factors (1 and n), then the number, n is not prime because it was divisible by other numbers.

Task 27:
Write **javacode** of a program that finds and displays all the prime numbers less than 1000.

Task 28:
Modify Task 26, instead of counting factors, print sum of factors.

Task 29:
Modify Task 26, calculate sum of factors less than the number itself. If the sum equals to the number, then print that the number is a perfect number.
Example 1: User enters n = 6. Factors of 6 are 1, 2, 3. Sum of those factors 1+2+3=6 which is same as the number 6(n). So, print that 6 is a perfect number.
Example 2: If user enters 8. Factors of 8 are 1, 2, 4. Sum of those factors 1+2+4=7 (NOT equal to 8).
So, print that 8 is NOT a perfect number.

Task 30:
Ask user for a range. Count how many numbers are prime number and how many numbers are perfect numbers between that range.
For example, between 2 and 6 there are 3 prime numbers (2, 3, 5) and 1 perfect number (6).

Sample Input:
2 6
Sample Output:
Between 2 and 6,
Found 3 prime numbers
Found 1 perfect number.