Task 01

[A,B,C,D should be written in a single java file]

A. Write a method called **evenChecker** that takes an **integer** number as its argument and prints whether the number is even or odd **inside the method**.

Sample Method Call	**Sample Output**
`evenChecker(10);`	`Even!!`
`evenChecker(17);`	`Odd!!`

B. Write a method called **isEven** that takes an **integer** number as an argument and **returns** boolean true if the number is even otherwise **returns** boolean false.

Sample Method Call	**Sample Output**

`boolean result = isEven(10);`
`System.out.println( result );`
	`true`

`boolean result = isEven(17);`
`System.out.println( result );`
	`false`

C. Write a method called **isPos** that takes an **integer** number as an argument and **returns** boolean true if the number is positive otherwise **returns** boolean false.

Sample Method Call	Sample Output

`boolean result = isPos(-5);`
`System.out.println( result );`
	`false`

`boolean result = isPos(12);`
`System.out.println( result );`
	`true`

D. Write a method called **sequence()** that takes an **integer** in its parameter called n. Now, if n is **positive** then it prints all the **even** numbers from **0 to n**, otherwise if n is **negative** it prints all the **odd** numbers from **n to -1**.

Note: **You must call** the methods from **1B** and **1C**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output	Explanation
`sequence(10);`	`0 2 4 6 8 10`	Here, 10 is positive so 0,2,4,6,8,10 were printed.
`sequence(-7);`	`-7 -5 -3 -1`	Here, -7 is negative so -7,-5,-3,-1 were printed.
`sequence(7);`	`0 2 4 6`	Here, 7 is positive so 0,2,4,6 were printed
`sequence(-8);`	`-7 -5 -3 -1`	Here, -8 is negative so -7,-5,-3,-1 were printed.

Task02

[A,B,C should be written in a single java file]

A. Write a method called **circleArea** that takes an **integer** radius in its parameter and **returns** the **area** of the circle.

Note: area of a circle is πr^2

Sample Method Call	Sample Output

`double area = circleArea(5);`
`System.out.println(area);`
	`78.5398`

B. Write a method called **sphereVolume** that takes an **integer** radius in its parameter and **returns** the **volume** of the sphere.

Note: volume of a sphere is $\frac{4}{3}\pi r^3$

Sample Method Call	Sample Output

`double volume = sphereVolume(5);`
`System.out.println(volume);`
	`523.5987`

C. Write a method called **findSpace** that takes two values in its parameters one is an **integer** diameter and another one is a String. Using the given diameter, this method should calculate the Area of a circle or the Volume of a sphere depending on the value of the second parameter. Finally, it should print the result **inside the method**.

Note: **You must call** the method written in task **2A & 2B**, otherwise this task would be **considered invalid**.

Sample Method Call	**Sample Output**
`findSpace(10,"circle");`	`78.5398`
`findSpace(5,"sphere");`	`33.5103`
`findSpace(10,"square");`	`Wrong Parameter`

Task03
[A,B should be written in a single java file]

A. Write a method called **isTriangle** that takes 3 integer numbers as arguments. The method will **return** the boolean True if the 3 sides can form a valid triangle otherwise it'll **return** the boolean False.

Note: In a valid triangle, the sum of **any** two sides will be greater than the third side.

Sample Method Call	Sample Output	Explanation

`boolean res = isTriangle(7,5,10);`
`System.out.println( res );`
	`true`	Here, 7+5>10, 5+10>7 also, 10+7>5. Thus, these 3 sides can form a valid triangle.

`boolean res = isTtriangle(3,2,1);`
`System.out.println( res );`
	`false`	Here, 1+2<=3, thus, these 3 sides can NOT form a valid triangle.

B. Write a method called **triArea** that takes 3 sides of a triangle as 3 **integer** arguments. The method should calculate and print the area of the triangle only if it's a valid triangle otherwise print that it's not a valid triangle.
Area of triangle = √[s(s−a)(s−b)(s−c)], where 's' is the semi perimeter of the triangle. So, semi-perimeter = s = perimeter/2 = (a + b + c)/2.

Note: **You must call** the method written in task **3A**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output	Explanation
`triArea(3,2,1);`	`Can't form triangle`	Here, 1+2<=3, thus, these 3 sides can NOT form a valid triangle.
`triArea(7,5,10);`	`16.248`	Here, 7,5,10 is able to form a valid triangle so, using the formula we get the area as 16.248

Task04

[A,B,C should be written in a single java file]

A. Write a method called **isPrime** which takes an integer in its parameter to check whether a number is prime or not. If the number is prime then the method returns boolean **true** otherwise it returns boolean **false**.

Sample Input	Sample Output
`boolean check = isPrime(7);` `System.out.println(check);`	true
`boolean check = isPrime(15);` `System.out.println(check);`	false

B. Write a method called **isPerfect** which takes an integer in its parameter to check whether a number is perfect or not. If the number is perfect then the method returns boolean **true** otherwise it returns boolean **false**.

Sample Input	Sample Output
`boolean check = isPerfect(6);` `System.out.println(check);`	true
`boolean check = isPerfect(33);` `System.out.println(check);`	false

C. Write a method called **special_sum** that calculates the sum of all numbers that are either prime numbers or perfect up till the integer value given in its parameter. This integer value must be taken as user input and passed into the method.

Note: You must call the methods written in task **4A & 4B**, otherwise this task will be considered invalid.

Sample Input	Sample Output	Output
8	`int result = special_sum(8);` `System.out.println(result);`	23
Explanation:	Between 1 to 8 the Prime numbers are 2,3,5,7 and 6 is a Perfect number. So, the summation is 2+3+5+7+6=23.	

Task05
[A,B,C should be written in a single java file]

A. Write a simple method called **showDots** that takes a number as an argument and then prints that amount of dots inside the method.

Note: You can use `System.out.print()` to avoid the next output being printed on the next line.

Sample Method Call	Sample Output
`showDots(5);`	.....
`showDots(3);`	...

B. Write a method called **show_palindrome** that takes a number as an argument and then prints a palindrome inside the method.

Note: You can use `System.out.print()` to avoid the next output being printed on the next line

Sample Method Call	Sample Output
`show_palindrome(5)`	123454321
`show_palindrome(3)`	12321

C. Write a method called **showDiamond** that takes an integer number as an argument and then prints a **palindromic diamond shape**. Moreover, the empty spaces surrounding the diamonds are filled with dots(**.**) .

Note: You must call the methods written in task **5A & 5B**, otherwise this task would be **considered invalid**.

Sample Method Call	Sample Output

`showDiamond(5)`	`....1....`
	`...121...`
	`..12321..`
	`.1234321.`
	`123454321`
	`.1234321.`
	`..12321..`
	`...121...`
	`....1....`

`showDiamond(3)`	`..1..`
	`.121.`
	`12321`
	`.121.`
	`..1..`

Task06

[A,B should be written in a single java file]

A. Write a method called **calcTax** that takes 2 arguments which are **your age** then **your salary**. The method must calculate and **return** the tax as per the following conditions:

- No tax if you are less than 18 years old.
- No tax if you get paid less than 10,000
- 7% tax if you get paid between 10K and 20K
- 14% tax if you get paid more than 20K

Sample Method Call	Output	Explanation
`double t = calcTax(16,20000);` `System.out.println(t);`	`0.0`	Here, the age is less than 18 so 0 tax.
`double t = calcTax(20,18000);` `System.out.println(t);`	`1260.0`	Here, the age is greater than 18 and income is between 10K-20K so tax is 7% of 18000 = 1260.

B. Write a method called **calcYearlyTax** that takes no arguments. Inside the method it should take **first input as your age** and then **12 other inputs** as income of each month of the year. The method must calculate and print Tax for each month and finally print the total Tax of the whole year based on the **6A conditions**.

Note: **You must call** the method written in task **6A**, otherwise this task would be **considered invalid**.

Sample Method Call	Input	Explanation

calcYearlyTax()	22
	8000
	15000
	22000
	2300
	15300
	21000
	34000
	9000
	27000
	88000
	32000
	7300
		`Month1 tax: 0`
		`Month2 tax: 1050.0`
		`Month3 tax: 3080.0`
		`Month4 tax: 0`
		`Month5 tax: 1071.0`
		`Month6 tax: 2940.0`
		`Month7 tax: 4760.0`
		`Month8 tax: 0`
		`Month9 tax: 3780.0`
		`Month10 tax: 12320.0`
		`Month11 tax: 4480.0`
		`Month12 tax: 0`
		`Total Yearly Tax: 33481.0`

Task07
[A,B,C should be written in a single java file]

A. Write a function called **oneToN** that prints 1 till N recursively.

Hint: N is a number taken as input from the user and you need to print the numbers starting from 1 to N recursively.

Sample Input	Sample Function Call	Output
N=5	oneToN(1,N);	1 2 3 4 5
N=11	oneToN(1,N);	1 2 3 4 5 6 7 8 9 10 11

B. Write a function **nToOne** that prints from N to 1 recursively.

Hint: N is a number taken as input from the user and you need to print the numbers starting from N to 1.

Sample Input	Sample Function Call	Output
N=6	nToOne(1,N);	6 5 4 3 2 1
N=3	nToOne(1,N);	3 2 1

C. Write a function called **recursiveSum** to sum till N recursively.

Hint: N is a number taken as input from the user and you need to add the numbers starting from 1 to N recursively and print the sum.

Sample Input	Sample Function Call	Output
N=4	recursiveSum(1,N);	10
N=12	recursiveSum(1,N);	78

Task08

Write a **recursive function** called **reverseDigits** that takes an integer n as an argument and prints the digits of n in reverse order.

Hint: Think about how you solved it using loop

Sample Input	Sample Function Call	Output
12345	reverseDigits(n)	5 4 3 2 1
649	reverseDigits(n)	9 4 6
1000	reverseDigits(n)	0 0 0 1

Task09

Write a **recursive function** called **sumDigits** that takes an integer n as an argument and sums up the digits of n then **returns** the result.

Hint: Think about how you would solve it using loop

Sample Input	Sample Function Call	Output

12345	int x = sumDigits(n);
	System.out.println(x);
		15

649	int x = sumDigits(n);
	System.out.println(x);
		19

Task 2

Write a method **modifyStrings()** that takes in three given strings **S, S1, and S2** consisting of different numbers of characters respectively, the task is to modify the string **S** by **replacing** all the **substrings S1** with the **string S2** in the string **S** and printing the modified string S.

Sample Input	Sample Output	Explanation

S = "abababa"
S1 = "aba"
S2 = "a"
 	
modifyStrings(S, S1, S2);
	aba	Changing the substrings **S[0, 2]**(Referring to characters from the 0th index of **S** till the 2nd index of S and **S[4, 6]** (= S1) to the string S2 (= "a") modifies the string S to "aba". Therefore, print "aba".

S = "baddadda"
S1 = "dd"
S2 = "n"
 	
modifyStrings(S, S1, S2);
	banana	Changing the substrings **S[2,3]**(Referring to characters from the 2nd and 3rd index of **S**) and **S[5, 6] (= S1)** to the string **S2 (= "n")** modifies the string S to "banana". Therefore, print "banana".

Task 4

Write a method called isHappyNumber which takes an integer in its parameter to check whether a number is a happy number or not. If the number is a happy number then the method returns boolean true otherwise it returns boolean false. In number theory, a happy number is a number which eventually reaches 1 when replaced by the sum of the square of each digit. For instance, 13 is a happy number because $1^2 + 3^2 = 10$ and $1^2 + 0^2 = 1$. On the other hand, 4 is not a happy number because the process continues in an infinite cycle without ever reaching 1. Unhappy number ends in a cycle of repeating numbers which contains 4 .

Sample Input	Sample Output
boolean check = isHappyNumber(82) System.out.println(check)	true
boolean check = isHappyNumber(4) System.out.println(check)	false

Task 5

Write a method called toDecimal which takes a binary number as a string in its parameter to convert the binary number to its decimal number and return the decimal value. After returning the decimal value, write another method called toHex which takes the converted decimal value in its parameter and calculates the hexadecimal value and then return the hex value.

Sample Input	Sample Output
int decimal = toDecimal("1010") String hex = toHex(decimal) System.out.println(hex)	"A"

Task 9

Your professor expects only As, Bs, and Cs. In the following program, write a method called **getScores** that takes as input corresponding arrays **studentGrades** and **studentScores**. Write a method called **getScores** that assigns **index i** in studentGrades based on **index i** in **studentScores**. If a grade is **A**, assign **100**. If a grade is **B**, assign **90**. If a grade is **C**, assign **70**. If a grade is anything else, assign **0**.

Sample Input	Sample Output

char[] studentGrades = new char[]{'A', 'A', 'A', 'B', 'C', 'U', 'Z'};
int[] studentScores = new int[7];
	Output expectation:
	100
	100
	100
	90
	70
	0
	0

Task 10

A. Write a method called **convertToCm()**, that takes as input a **type double** and **returns** the value converted from inches to centimeters.
Hint: There are 2.54 centimeters in an inch

Sample Method Call	Output

double t = convertToCm(16);
System.out.println(t + " cm");
	40.64 cm

B. Create an **array** of **type double** of length **5** called **cheetos_inches**, that stores the length of each of the Cheetos **from the user**. Send the array of length in inches into a method called **findAvgCm()** that **returns** the average length of the Cheetos **in cm to 2 decimal places**. The method findAvgCm() uses **convertTocm()** to convert the length of each Cheetos **from inches to cm.**
Note: You must call the method written in [Method Task A], otherwise this task would be considered invalid.

Sample Method Call	Output

Sample array:
double [] cheetos_inches = new double[]{10.0, 12.0, 14.0, 16.0, 18.0};
 	
averageLength = findAvgCm(cheetos_inches);
 	
System.out.println("The average Cheeto length is "+ averageLength +" cm");
	The average Cheeto length is 35.56 cm

Task 11

A. Write a method called **isVowel** which takes a string in its parameter and counts all the vowels in the String. If any vowel exists in the string then the method returns the **count**.

Sample Input	Sample Output
The quick brown fox jumps over the lazy dog	Number of vowels in the string: 11

B. Write a method called **isConsonant** which takes a string in its parameter and counts all the consonants in the String. If any consonant exists in the string then the method returns the **count**.

Sample Input	Sample Output
The quick brown fox jumps over the lazy dog	Number of consonants in the string: 24

C. Write a method called **vowel/consonantSum** which takes an array of strings in its parameter and returns the summation of the number of vowels/consonants.
Note: You must call the methods written in tasks **A/B**, otherwise this task will be **considered invalid**.

Given Array	Sample Output
String [] names = {"Bob", "Alice", "Max", "Marry", "Rosy"}; System.out.println("The total number of vowels in the array is:" + vowelSum(names)); System.out.println("The total number of consonants in the array is:" + consonantSum(names));	The total number of vowels in the array is: **7** The total number of consonants in the array is: **13**