**Assignment Recursion**

Q1. Q1 : Given an integer, find out the sum of its digits using recursion.

Input: n= 1234

Output: 10

Explanation: 1+2+3+4=10

public class SumOfDigits {

// Recursive method to find the sum of digits

public static int sumOfDigits(int n) {

// Base case: when n is a single digit

if (n == 0) {

return 0;

}

// Recursive case: add the last digit and call the method with the remaining digits

return n % 10 + sumOfDigits(n / 10);

}

public static void main(String[] args) {

int number = 1234;

int result = sumOfDigits(number);

System.out.println("The sum of the digits is: " + result);

}

}

Q2: Given a number n. Find the sum of natural numbers till n but with alternate signs.

That means if n = 5 then you have to return 1-2+3-4+5 = 3 as your answer.

Constraints : 0<=n<=1e6

Input1 : n = 10

Output 1 : -5

Explanation : 1-2+3-4+5-6+7-8+9-10 = -5

Input 2 : n = 5

Output 2 : 3

public class AlternatingSum {

// Method to calculate the sum of natural numbers with alternating signs

public static int alternatingSum(int n) {

int sum = 0;

for (int i = 1; i <= n; i++) {

if (i % 2 == 0) {

sum -= i; // Subtract if the number is even

} else {

sum += i; // Add if the number is odd

}

}

return sum;

}

public static void main(String[] args) {

int number1 = 10;

int number2 = 5;

System.out.println("The alternating sum up to " + number1 + " is: " + alternatingSum(number1));

System.out.println("The alternating sum up to " + number2 + " is: " + alternatingSum(number2));

}

}

Q3: Print the max value of the array [ 13, 1, -3, 22, 5]

public class MaxValueFinder {

// Method to find the maximum value in an array

public static int findMax(int[] array) {

int max = array[0]; // Initialize max with the first element of the array

for (int i = 1; i < array.length; i++) {

if (array[i] > max) {

max = array[i]; // Update max if the current element is greater

}

}

return max;

}

public static void main(String[] args) {

int[] array = {13, 1, -3, 22, 5};

int maxValue = findMax(array);

System.out.println("The maximum value in the array is: " + maxValue);

}

}

Q4 : Find the sum of the values of the array [92, 23, 15, -20, 10].

public class ArraySum {

// Method to find the sum of values in an array

public static int calculateSum(int[] array) {

int sum = 0; // Initialize sum to 0

for (int value : array) {

sum += value; // Add each element to sum

}

return sum;

}

public static void main(String[] args) {

int[] array = {92, 23, 15, -20, 10};

int totalSum = calculateSum(array);

System.out.println("The sum of the array values is: " + totalSum);

}

}

Q5. Given a number n. Print if it is an armstrong number or not. An armstrong number is a number if the sum

of every digit in that number raised to the power of total digits in that number is equal to the number.

Example : 153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153 hence 153 is an armstrong number. (Easy)

Input1 : 153

Output1 : Yes

Input 2 : 134

Output2 : No

public class ArmstrongNumber {

// Method to check if a number is an Armstrong number

public static boolean isArmstrongNumber(int n) {

int originalNumber = n;

int numberOfDigits = String.valueOf(n).length();

int sum = 0;

while (n > 0) {

int digit = n % 10; // Extract the last digit

sum += Math.pow(digit, numberOfDigits); // Raise digit to the power of numberOfDigits and add to sum

n /= 10; // Remove the last digit

}

return sum == originalNumber; // Check if sum equals the original number

}

public static void main(String[] args) {

int number1 = 153;

int number2 = 134;

System.out.println(number1 + " is an Armstrong number: " + (isArmstrongNumber(number1) ? "Yes" : "No"));

System.out.println(number2 + " is an Armstrong number: " + (isArmstrongNumber(number2) ? "Yes" : "No"));

}

}