Q1.public class PowerOfTwo {

public static boolean isPowerOfTwo(int n) {

if (n <= 0) {

return false;

}

return (n & (n - 1)) == 0;

}

public static void main(String[] args) {

System.out.println(isPowerOfTwo(1)); // Output: true

System.out.println(isPowerOfTwo(16)); // Output: true

System.out.println(isPowerOfTwo(3)); // Output: false

}

}

Q2.public class SumOfNaturalNumbers {

public static int sumOfFirstNNumbers(int n) {

int sum = 0;

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

sum += i;

}

return sum;

}

public static void main(String[] args) {

System.out.println(sumOfFirstNNumbers(3)); // Output: 6

System.out.println(sumOfFirstNNumbers(5)); // Output: 15

System.out.println(sumOfFirstNNumbers(10)); // Output: 55

}

}

Q3.public class Factorial {

public static int factorial(int n) {

int fact = 1;

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

fact \*= i;

}

return fact;

}

public static void main(String[] args) {

System.out.println(factorial(5)); // Output: 120

System.out.println(factorial(4)); // Output: 24

System.out.println(factorial(10)); // Output: 3628800

}

}

Q4.public class Exponent {

public static int power(int N, int P) {

int result = 1;

for (int i = 0; i < P; i++) {

result \*= N;

}

return result;

}

public static void main(String[] args) {

System.out.println(power(5, 2)); // Output: 25

System.out.println(power(2, 5)); // Output: 32

System.out.println(power(10, 3)); // Output: 1000

}

}

Q5.public class MaximumElement {

public static int findMax(int[] arr, int start, int end) {

// Base case: If there's only one element in the array, return that element

if (start == end) {

return arr[start];

}

// Find the maximum element in the remaining subarray recursively

int mid = (start + end) / 2;

int leftMax = findMax(arr, start, mid);

int rightMax = findMax(arr, mid + 1, end);

// Return the maximum of leftMax and rightMax

return Math.max(leftMax, rightMax);

}

public static void main(String[] args) {

int[] arr = {1, 4, 3, -5, -4, 8, 6};

int max = findMax(arr, 0, arr.length - 1);

System.out.println("Maximum element: " + max); // Output: 8

int[] arr2 = {1, 4, 45, 6, 10, -8};

int max2 = findMax(arr2, 0, arr2.length - 1);

System.out.println("Maximum element: " + max2); // Output: 45

}

}

Q6.public class ArithmeticProgression {

public static int findNthTerm(int a, int d, int N) {

// Calculate the Nth term using the formula: a + (N - 1) \* d

int nthTerm = a + (N - 1) \* d;

return nthTerm;

}

public static void main(String[] args) {

int a1 = 2, d1 = 1, N1 = 5;

int nthTerm1 = findNthTerm(a1, d1, N1);

System.out.println("Nth term: " + nthTerm1); // Output: 6

int a2 = 5, d2 = 2, N2 = 10;

int nthTerm2 = findNthTerm(a2, d2, N2);

System.out.println("Nth term: " + nthTerm2); // Output: 23

}

}

Q7.import java.util.\*;

public class Permutations {

public static void printPermutations(String str) {

// Convert the string to a character array for easier manipulation

char[] chars = str.toCharArray();

// Create a list to store the permutations

List<String> permutations = new ArrayList<>();

// Generate permutations using backtracking

generatePermutations(chars, 0, permutations);

// Print the permutations

for (String permutation : permutations) {

System.out.println(permutation);

}

}

private static void generatePermutations(char[] chars, int index, List<String> permutations) {

if (index == chars.length - 1) {

// Base case: Add the current permutation to the list

permutations.add(String.valueOf(chars));

} else {

// Generate permutations by swapping each character with the current index

for (int i = index; i < chars.length; i++) {

swap(chars, index, i);

generatePermutations(chars, index + 1, permutations);

swap(chars, index, i); // Backtrack: Swap back to the original order

}

}

}

private static void swap(char[] chars, int i, int j) {

char temp = chars[i];

chars[i] = chars[j];

chars[j] = temp;

}

public static void main(String[] args) {

String str1 = "ABC";

System.out.println("Permutations of " + str1 + ":");

printPermutations(str1);

String str2 = "XY";

System.out.println("Permutations of " + str2 + ":");

printPermutations(str2);

}

}

Q8.public class ArrayProduct {

public static int findProduct(int[] arr) {

int product = 1;

for (int i = 0; i < arr.length; i++) {

product \*= arr[i];

}

return product;

}

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 4, 5};

int product1 = findProduct(arr1);

System.out.println("Product of elements in arr1: " + product1);

int[] arr2 = {1, 6, 3};

int product2 = findProduct(arr2);

System.out.println("Product of elements in arr2: " + product2);

}

}