Home Tasks:

Recursion:

Wap to find the factorial of a number

package worked\_codes.July4;  
  
import java.util.Scanner;  
  
public class FactorialExample {  
  
 // Iterative method  
 public static long factorialIterative(int n) {  
 long fact = 1;  
 for (int i = 1; i <= n; i++) {  
 fact \*= i;  
 }  
 return fact;  
 }  
  
 // Recursive method  
 public static long factorialRecursive(int n) {  
 if (n == 0 || n == 1)  
 return 1;  
 return n \* *factorialRecursive*(n - 1);  
 }  
  
 // Main method  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a non-negative integer: ");  
 int num = scanner.nextInt();  
  
 if (num < 0) {  
 System.*out*.println("Factorial is not defined for negative numbers.");  
 } else {  
 System.*out*.println("Factorial (Iterative) of " + num + " is: " + *factorialIterative*(num));  
 System.*out*.println("Factorial (Recursive) of " + num + " is: " + *factorialRecursive*(num));  
 }  
  
 scanner.close();  
 }  
}

output:

Enter a non-negative integer: 52

Factorial (Iterative) of 52 is: -8452693550620999680

Factorial (Recursive) of 52 is: -8452693550620999680

Process finished with exit code 0

Wap to find the Fibonacci series of a number

package worked\_codes.July4;  
  
import java.util.Scanner;  
  
public class FibonacciSeries {  
  
 // Method to print Fibonacci series up to n terms  
 public static void printFibonacci(int n) {  
 int first = 0, second = 1;  
  
 System.*out*.print("Fibonacci series up to " + n + " terms: ");  
  
 for (int i = 1; i <= n; i++) {  
 System.*out*.print(first + " ");  
 int next = first + second;  
 first = second;  
 second = next;  
 }  
  
 System.*out*.println();  
 }  
  
 // Main method  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter number of terms: ");  
 int num = scanner.nextInt();  
  
 if (num <= 0) {  
 System.*out*.println("Please enter a positive number.");  
 } else {  
 *printFibonacci*(num);  
 }  
  
 scanner.close();  
 }  
}

output:

Enter number of terms: 12

Fibonacci series up to 12 terms: 0 1 1 2 3 5 8 13 21 34 55 89

Process finished with exit code 0

What is the difference between recursion and iteration

| **Feature** | **Iteration** | **Recursion** |
| --- | --- | --- |
| **Definition** | Repeats a block of code using loops (for, while) | Function calls itself to solve smaller subproblems. |
| **Control Structure** | Uses **loops** | Uses **function calls** |
| **Termination** | Controlled by loop condition | Controlled by base case |
| **Performance** | Generally faster and uses less memory | Slower due to function call overhead and stack usage |
| **Memory Use** | Uses a fixed amount of memory | Uses **call stack**, can lead to StackOverflowError |
| **Clarity** | May be more complex for mathematical problems | Often more elegant and concise for mathematical problems |
| **Example** | Fibonacci using loop | Fibonacci using self-calling function |

Wap to reverse a string using recursion..

package worked\_codes.July4;  
  
import java.util.Scanner;  
  
public class ReverseStringRecursive {  
  
 // Recursive method to reverse a string  
 public static String reverse(String str) {  
 if (str.isEmpty()) {  
 return str;  
 }  
  
 // Take the last character + reverse of the rest  
 return *reverse*(str.substring(1)) + str.charAt(0);  
 }  
  
 // Main method  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter a string to reverse: ");  
 String input = scanner.nextLine();  
  
 String reversed = *reverse*(input);  
 System.*out*.println("Reversed string: " + reversed);  
  
 scanner.close();  
 }  
}

output:

Enter a string to reverse: Jaysree

Reversed string: eersyaJ

Process finished with exit code 0

Write a recursive function to search for an element in an array

Code:

package worked\_codes.July4;  
  
public class RecursiveSearch {  
  
 // Recursive method to search for an element  
 public static int recursiveSearch(int[] arr, int target, int index) {  
 // Base case: reached end of array  
 if (index >= arr.length) {  
 return -1; // not found  
 }  
  
 // If element found  
 if (arr[index] == target) {  
 return index;  
 }  
  
 // Recursive call for next index  
 return *recursiveSearch*(arr, target, index + 1);  
 }  
  
 // Main method  
 public static void main(String[] args) {  
 int[] array = {5, 8, 12, 3, 7, 9};  
 int target = 3;  
  
 int result = *recursiveSearch*(array, target, 0);  
  
 if (result == -1) {  
 System.*out*.println("Element not found.");  
 } else {  
 System.*out*.println("Element found at index: " + result);  
 }  
 }  
}

output:

Enter a string to reverse: JAYSREE SANJAI

Reversed string: IAJNAS EERSYAJ

Process finished with exit code 0

Write a recursive function to count the digits of a positive integer (do also for sum of digits)

package worked\_codes.July4;  
  
public class DigitOperations {  
  
 // Recursive method to count digits  
 public static int countDigits(int number) {  
 if (number == 0) {  
 return 0;  
 }  
 return 1 + *countDigits*(number / 10);  
 }  
  
 // Recursive method to sum digits  
 public static int sumDigits(int number) {  
 if (number == 0) {  
 return 0;  
 }  
 return (number % 10) + *sumDigits*(number / 10);  
 }  
  
 // Main method  
 public static void main(String[] args) {  
 int num = 12345;  
  
 int digitCount = (num == 0) ? 1 : *countDigits*(num); // handle 0  
 int digitSum = *sumDigits*(num);  
  
 System.*out*.println("Number: " + num);  
 System.*out*.println("Digit count: " + digitCount);  
 System.*out*.println("Sum of digits: " + digitSum);  
 }  
}

output:

Number: 12345

Digit count: 5

Sum of digits: 15

Process finished with exit code 0

Write a recursive function to reverse a null-terminated string package worked\_codes.July4;  
  
public class ReverseNullTerminated {  
  
 // Recursive function to reverse a string until '\0'  
 public static void reverseString(char[] str, int index) {  
 // Base case: if null terminator or end of array  
 if (index >= str.length || str[index] == '\0') {  
 return;  
 }  
  
 // Recursive call  
 *reverseString*(str, index + 1);  
  
 // Print after recursive call (reversing order)  
 System.*out*.print(str[index]);  
 }  
  
 public static void main(String[] args) {  
 // Simulating a null-terminated string (note the '\0' at the end)  
 char[] input = {'H', 'e', 'l', 'l', 'o', '\0'};  
  
 System.*out*.print("Reversed string: ");  
 *reverseString*(input, 0);  
 System.*out*.println();  
 }  
}

Output:

Reversed string: olleH

Process finished with exit code 0

Write a recursive function to convert a decimal number to binary

package worked\_codes.July4;  
  
public class DecimalToBinary {  
  
 // Recursive function to convert decimal to binary  
 public static void toBinary(int number) {  
 if (number == 0) {  
 return;  
 }  
  
 *toBinary*(number / 2); // Recursive call  
 System.*out*.print(number % 2); // Print binary digit  
 }  
  
 public static void main(String[] args) {  
 int decimal = 13;  
  
 if (decimal == 0) {  
 System.*out*.print("0");  
 } else {  
 System.*out*.print("Binary of " + decimal + " is: ");  
 *toBinary*(decimal);  
 }  
  
 System.*out*.println(); // for newline  
 }  
}

code:

Binary of 13 is: 1101

Process finished with exit code 0

 Write a recursive function to check if a string is a palindrome or not

package worked\_codes.July4;  
  
public class PalindromeRecursive {  
  
 // Recursive method to check for palindrome  
 public static boolean isPalindrome(String str, int left, int right) {  
 // Base case: crossed or single character  
 if (left >= right) {  
 return true;  
 }  
  
 // If mismatch found  
 if (str.charAt(left) != str.charAt(right)) {  
 return false;  
 }  
  
 // Recursive call: move inward  
 return *isPalindrome*(str, left + 1, right - 1);  
 }  
  
 public static void main(String[] args) {  
 String input = "madam";  
  
 boolean result = *isPalindrome*(input, 0, input.length() - 1);  
  
 if (result) {  
 System.*out*.println(input + " is a palindrome.");  
 } else {  
 System.*out*.println(input + " is not a palindrome.");  
 }  
 }  
}

output:

madam is a palindrome.

Process finished with exit code 0

Write a recursive function to copy one array to another

package worked\_codes.July4;  
  
public class ArrayCopyRecursive {  
  
 // Recursive method to copy elements from source[] to destination[]  
 public static void copyArray(int[] source, int[] destination, int index) {  
 // Base case: if end of array is reached  
 if (index >= source.length) {  
 return;  
 }  
  
 // Copy current element  
 destination[index] = source[index];  
  
 // Recursive call for next index  
 *copyArray*(source, destination, index + 1);  
 }  
  
 public static void main(String[] args) {  
 int[] source = {10, 20, 30, 40, 50};  
 int[] destination = new int[source.length];  
  
 *copyArray*(source, destination, 0);  
  
 // Print destination array  
 System.*out*.print("Copied array: ");  
 for (int num : destination) {  
 System.*out*.print(num + " ");  
 }  
 }  
}

output:

Copied array: 10 20 30 40 50

Process finished with exit code 0