DOMAIN WINTER CAMP

(Department of Computer Science and Engineering)

Name: Shreyansh Kumar UID: 22BCS11674 Section: ML-904

DAY 3

Ques 1. **Fibonnacci Series Using Recursion**

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1

F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

Program code:

```
#include <iostream>
using namespace std;
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    }
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

```
int main() {
    int n;
    cout << "Enter the number of terms in the Fibonacci series: ";
    cin >> n;

    cout << "Fibonacci series: ";
    for (int i = 0; i < n; i++) {
        cout << fibonacci(i) << " ";
    }
    cout << endl;

    return 0;
}
```

Output:

```
Enter the number of terms in the Fibonacci series: 6
Fibonacci series: 0 1 1 2 3 5
```

Ques 2. **Factorial Of Number Using Recursion**

Write a program that returns the value of N! (N factorial) using recursion.
Note that N! =- 1*2*...*N
Also, 0! = 1 and 1! = 1.

Program code:

```
#include <iostream>
using namespace std;


int factorial(int n) {
    if (n <= 1) {
        return 1;
    }
```

```
    return n * factorial(n - 1);
}

int main() {
    int num;
    cout << "Enter a number to calculate its factorial: ";
    cin >> num;

    if (num < 0) {
        cout << "Factorial of a negative number is not defined." << endl;
    } else {
        cout << "Factorial of " << num << " is: " << factorial(num) << endl;
    }

    return 0;
}
```

Output:

```
Enter a number to calculate its factorial: 9
Factorial of 9 is: 362880
```

Ques 3. **Sum of Natural Number Using Recursion**

Given a number n, find sum of first n natural numbers. To calculate the sum, we will use a recursive function recur_sum().

Program Code:

```
#include <iostream>

using namespace std;


// Recursive function to calculate the sum of natural numbers
```

```
int sumOfNaturalNumbers(int n) {
    if (n == 0)
        return 0; // Base case: sum of 0 is 0
    return n + sumOfNaturalNumbers(n - 1); // Recursive case
}

int main() {
    int n;
    // Input from the user
    cout << "Enter a positive integer: ";
    cin >> n;
    if (n < 0) {
        cout << "Please enter a positive integer." << endl;
    } else {
        // Calculate the sum using recursion
        int result = sumOfNaturalNumbers(n);
        // Output the result
        cout << "The sum of natural numbers up to " << n << " is: " << result <<
endl;
    }
    return 0;
}
```

Output:

```
Enter a positive integer: 5
The sum of natural numbers up to 5 is: 15
```

Ques 4. **Sum of Array Elements Using Recursion**

Given an array of integers, find sum of array elements using recursion.

Program Code:

```
#include <iostream>
using namespace std;

// Recursive function to calculate the sum of array elements
int sumOfArray(int arr[], int size) {
    if (size == 0)
        return 0; // Base case: sum of an empty array is 0
    return arr[size - 1] + sumOfArray(arr, size - 1); // Recursive case
}

int main() {
    int n;

    // Input size of the array
    cout << "Enter the number of elements in the array: ";
```

```
    cin >> n;

    if (n <= 0) {
        cout << "Array size must be positive." << endl;
        return 0;
    }

    int arr[n];

    // Input array elements
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Calculate the sum using recursion
    int result = sumOfArray(arr, n);

    // Output the result
    cout << "The sum of the array elements is: " << result << endl;

    return 0;
}
```

Output:

```
Enter the number of elements in the array: 5
Enter the elements of the array: 3 4 5 2 6
The sum of the array elements is: 20
```

Ques 5. **Merge Two Sorted Lists**
You are given the heads of two sorted linked lists list1 and list2.
Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.
Return the head of the merged linked list.

Program Code:

```
#include <iostream>
using namespace std;

// Definition for a singly linked list node
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};

// Function to merge two sorted lists
ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
    // Base cases
    if (!l1) return l2;
    if (!l2) return l1;

    // Recursive merge
    if (l1->val < l2->val) {
        l1->next = mergeTwoLists(l1->next, l2);
        return l1;
    } else {
        l2->next = mergeTwoLists(l1, l2->next);
        return l2;
    }
```

```
}

// Function to print the linked list
void printList(ListNode* head) {
    while (head) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

// Helper function to insert a new node at the end of the list
ListNode* insertAtEnd(ListNode* head, int val) {
    if (!head) return new ListNode(val);
    ListNode* temp = head;
    while (temp->next) temp = temp->next;
    temp->next = new ListNode(val);
    return head;
}

int main() {
    // Create two sorted linked lists
    ListNode* l1 = nullptr;
    ListNode* l2 = nullptr;

    int n1, n2, val;

    cout << "Enter the number of elements in the first sorted list: ";
    cin >> n1;
    cout << "Enter elements of the first sorted list: ";
    for (int i = 0; i < n1; i++) {
        cin >> val;
        l1 = insertAtEnd(l1, val);
    }

    cout << "Enter the number of elements in the second sorted list: ";
    cin >> n2;
    cout << "Enter elements of the second sorted list: ";
    for (int i = 0; i < n2; i++) {
        cin >> val;
```

```
        l2 = insertAtEnd(l2, val);
    }

    // Merge the two sorted lists
    ListNode* mergedList = mergeTwoLists(l1, l2);

    // Print the merged list
    cout << "Merged Sorted List: ";
    printList(mergedList);

    return 0;
}
```

Output:

```
Enter the number of elements in the first sorted list: 4
Enter elements of the first sorted list: 3 4 5 6
Enter the number of elements in the second sorted list: 4
Enter elements of the second sorted list: 5 6 7 8
Merged Sorted List: 3 4 5 5 6 6 7 8
```