Name: Shreya Shree
UID:22BCS10174

Section: KPIT_901

Day 3

Q1. Print Full Name Using Function

Code:

Day 3 > name.cpp > main()

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

void print_full_name(string first, string last) {
    cout << "Hello " << first << " " << last << "! You just delved into function." << endl;
}

int main() {
    string first, last;
    cout << "Enter first name: ";
    getline(cin, first);
    cout << "Enter last name: ";
    getline(cin, last);

    print_full_name(first, last);

    return 0;
}
```

Output Example:

```
PS C:\Users\ASUS\Downloads\mycode> cd day3
PS C:\Users\ASUS\Downloads\mycode\day3> g++ name.cpp -o name
PS C:\Users\ASUS\Downloads\mycode\day3> ./name
Enter first name: Shreya
Enter last name: Shree
Hello Shreya  Shree! You just delved into function.
```

Q2 Find GCD of Number Using Function

Given an integer array nums, return the greatest common divisor of the smallest number and largest number in nums. The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers.

Day3 > GCD.cpp > main()

```cpp
#include <iostream>

using namespace std;

int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int findGCD(const vector<int>& nums) {
    int minNum = *min_element(nums.begin(), nums.end());
    int maxNum = *max_element(nums.begin(), nums.end());
    return gcd(minNum, maxNum);
}

int main() {
    vector<int> nums = {2, 5, 6, 9, 10};
    cout << "GCD: " << findGCD(nums) << endl;

    return 0;
}
```

Output:

```
PS C:\Users\ASUS\Downloads\mycode\day3> g++ GCD.cpp -o GCD
PS C:\Users\ASUS\Downloads\mycode\day3> ./GCD
GCD: 2
PS C:\Users\ASUS\Downloads\mycode\day3>
```

Q3: Longest Substring Without Repeating Characters

You are working on building a text editor application. One of the features you're focusing on is a "word suggestion" system. The editor needs to identify the longest sequence of characters typed without repeating any letters to suggest potential words or phrases. To accomplish this, you must efficiently find the length of the longest substring of unique characters as the user types.

Code:

Day3 > substring.cpp > main()

```cpp
#include <iostream>
#include <unordered_map>
#include <string>
using namespace std;

int lengthOfLongestSubstring(string s) {
    unordered_map<char, int> charIndexMap;
    int maxLength = 0;
    int start = 0;

    for (int i = 0; i < s.length(); i++) {
        if (charIndexMap.find(s[i]) != charIndexMap.end()) {
            start = max(start, charIndexMap[s[i]] + 1);
        }
        charIndexMap[s[i]] = i;
        maxLength = max(maxLength, i - start + 1);
    }
    return maxLength;
}

int main() {
    string input = "abcabcbb";
    cout << "Length of longest substring without repeating characters: " << lengthOfLongestSubstring(input) << endl;

    return 0;
}
```

Output:

```
PS C:\Users\ASUS\Downloads\mycode\day3> g++ substring.cpp -o substring
PS C:\Users\ASUS\Downloads\mycode\day3> ./substring
Length of longest substring without repeating characters: 3
PS C:\Users\ASUS\Downloads\mycode\day3>
```

Q 4.**Fibonacci 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).

Code:

Day3 > fibbonacci.cpp > ...

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

int fibonacci(int n) {
    if (n <= 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}

int main() {
    int result = fibonacci(6);
    cout << result << endl;
    return 0;
}

```

Output:

```
PS C:\Users\ASUS\Downloads\mycode\day3> g++ fibbonacci.cpp -o fibbonacci
PS C:\Users\ASUS\Downloads\mycode\day3> ./fibbonacci
8
PS C:\Users\ASUS\Downloads\mycode\day3>
```

Q5. Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Day3 > add.cpp > ...

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

struct ListNode {
    int val;
    ListNode *next;
    ListNode(int x) : val(x), next(NULL) {}
};

ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* dummy = new ListNode(0);
    ListNode* p = l1, *q = l2, *current = dummy;
    int carry = 0;

    while (p != NULL || q != NULL || carry) {
        int x = (p != NULL) ? p->val : 0;
        int y = (q != NULL) ? q->val : 0;
        int sum = carry + x + y;
        carry = sum / 10;
        current->next = new ListNode(sum % 10);
        current = current->next;
        if (p != NULL) p = p->next;
        if (q != NULL) q = q->next;
    }
    return dummy->next;
}

ListNode* createList(vector<int> values) {
    ListNode* head = new ListNode(values[0]);
    ListNode* current = head;
    for (int i = 1; i < values.size(); i++) {
        current->next = new ListNode(values[i]);
        current = current->next;
    }
    return head;
```

```
}

void printList(ListNode* head) {
    while (head) {
        cout << head->val << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    ListNode* l1 = createList({2, 4, 3});
    ListNode* l2 = createList({5, 6, 4});

    ListNode* result = addTwoNumbers(l1, l2);
    cout << "Sum List: ";
    printList(result);

    return 0;
}
```

Output:

```
PS C:\Users\ASUS\Downloads\mycode\day3> g++ add.cpp -o add
PS C:\Users\ASUS\Downloads\mycode\day3> ./add
Sum List: 7 0 8
PS C:\Users\ASUS\Downloads\mycode\day3>
```