WWC ASSIGNMENT DAY 3

NAME: Shreya Biswas **UID: 22BCS15325**
DATE: 23 DEC 2024 **SEC:620 A**

Question -- write a function to add two numbers

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

// Function to add two numbers
int addNumbers(int a, int b) {
    return a + b;
}

int main() {
    int num1, num2;

    // Input two numbers
    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;

    // Call the function and display the result
    int sum = addNumbers(num1, num2);
    cout << "The sum of " << num1 << " and " << num2 << " is: " << sum << endl;

    return 0;
}
```

Question-- create a function prime or not

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

bool isPrime(int n) {
    if (n <= 1)
        return false;
    for (int i = 2; i * i <= n; i++) {
```

```
        if (n % i == 0)
            return false;
    }
    return true;
}

int main() {
    int num;


    cout << "Enter a number: ";
    cin >> num;

    if (isPrime(num))
        cout << num << " is a prime number." << endl;
    else
        cout << num << " is not a prime number." << endl;

    return 0;
}
*/
```

Question-- create a function to check number is GCD

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

int gcd(int a, int b) {
    if (b == 0)
        return a;
    return gcd(b, a % b);
}

int main() {
    int num1, num2;

    cout << "Enter two numbers: ";
    cin >> num1 >> num2;


    int result = gcd(num1, num2);
```

```
    cout << "The GCD of " << num1 << " and " << num2 << " is: " << result <<
endl;

    return 0;
}


*/
```

Question-- Iterative C++ program to reverse a linked list

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

class Node {
public:
    int data;
    Node* next;

    Node(int new_data) {
        data = new_data;
        next = nullptr;
    }
};

Node* reverseList(Node* head) {


    Node *curr = head, *prev = nullptr, *next;


    while (curr != nullptr) {


        next = curr->next;


        curr->next = prev;


        prev = curr;
        curr = next;
    }
```

```
    return prev;
}

void printList(Node* node) {
    while (node != nullptr) {
        cout << " " << node->data;
        node = node->next;
    }
}

int main() {


    Node* head = new Node(1);
    head->next = new Node(2);
    head->next->next = new Node(3);
    head->next->next->next = new Node(4);
    head->next->next->next->next = new Node(5);

    cout << "Given Linked list:";
    printList(head);

    head = reverseList(head);

    cout << "\nReversed Linked List:";
    printList(head);

    return 0;
}
*/
```

Question-- Create a function to check number is perfect or not

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


bool isPerfectNumber(int n) {
    if (n <= 1) return false;

    int sum = 0;
    for (int i = 1; i <= n / 2; ++i) {
        if (n % i == 0) {
```

```
            sum += i;
        }
    }
    return sum == n;
}

int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;

    if (isPerfectNumber(n0umber)) {
        cout << number << " is a perfect number." << endl;
    } else {
        cout << number << " is not a perfect number." << endl;
    }

    return 0;
}
*/
```

Question -- reverse a string in c++

```
/*
#include <iostream>
#include <string>
using namespace std;


string reverseString(const string& input) {
    string reversed = input;
    int n = reversed.length();

    for (int i = 0; i < n / 2; ++i) {
        swap(reversed[i], reversed[n - i - 1]);
    }

    return reversed;
}

int main() {
    string str = "Hello, World!";
    cout << "Original String: " << str << endl;
    cout << "Reversed String: " << reverseString(str) << endl;
    return 0;
```

}
*/

Question-- write a c++ program to create a calulator to perform basic arthmetic operation

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

// Function prototypes
void displayMenu();
float add(float a, float b);
float subtract(float a, float b);
float multiply(float a, float b);
float divide(float a, float b);

int main() {
    float num1, num2, result;
    char operation;
    bool run = true;

    while (run) {
        displayMenu();
        cout << "Enter your choice (+, -, *, / or q to quit): ";
        cin >> operation;

        if (operation == 'q') {
            cout << "Exiting the calculator. Goodbye!" << endl;
            run = false;
            continue;
        }

        // Input numbers
        cout << "Enter the first number: ";
        cin >> num1;
        cout << "Enter the second number: ";
        cin >> num2;

        // Perform the operation
        switch (operation) {
            case '+':
                result = add(num1, num2);
                break;
```

```
            case '-':
                result = subtract(num1, num2);
                break;
            case '*':
                result = multiply(num1, num2);
                break;
            case '/':
                if (num2 == 0) {
                    cout << "Error: Division by zero is not allowed!" << endl;
                    continue;
                }
                result = divide(num1, num2);
                break;
            default:
                cout << "Invalid operation. Please try again!" << endl;
                continue;
        }

        cout << "The result is: " << result << endl;
    }

    return 0;
}


void displayMenu() {
    cout << "\nSimple Calculator" << endl;
    cout << "-----------------" << endl;
    cout << "Choose an operation:" << endl;
    cout << "+ : Addition" << endl;
    cout << "- : Subtraction" << endl;
    cout << "* : Multiplication" << endl;
    cout << "/ : Division" << endl;
    cout << "q : Quit" << endl;
}

float add(float a, float b) {
    return a + b;
}

float subtract(float a, float b) {
    return a - b;
}
```

```
float multiply(float a, float b) {
   return a * b;
}

float divide(float a, float b) {
   return a / b;
}
*/
```

Question-- Function to check if a number is a palindrome

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

bool isPalindrome(int number) {
   int original = number;
   int reversed = 0;

   while (number > 0) {
      int digit = number % 10;
      reversed = reversed * 10 + digit;
      number /= 10;
   }

   return original == reversed;
}

int main() {
   int num;


   cout << "Enter a number: ";
   cin >> num;


   if (isPalindrome(num)) {
      cout << num << " is a palindrome!" << endl;
   } else {
      cout << num << " is not a palindrome!" << endl;
   }

   return 0;
}
*/
```

Question -- Function to calculate the sum of array elements using recursion

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


int sumArray(int arr[], int size) {

   if (size == 0) {
      return 0;
   }
   return arr[size - 1] + sumArray(arr, size - 1);
}

int main() {
   int n;


   cout << "Enter the number of elements in the array: ";
   cin >> n;


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

   int result = sumArray(arr, n);


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

   return 0;
}
*/
```

Question -- reverse the node of list k at a time and return modified list

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

struct ListNode {
   int val;
   ListNode *next;
```

```
    ListNode(int x) : val(x), next(nullptr) {}
};


ListNode* reverseKGroup(ListNode* head, int k) {
    if (!head || k == 1) return head;

    ListNode* dummy = new ListNode(0);
    dummy->next = head;
    ListNode* prevGroupEnd = dummy;
    ListNode* current = head;

    while (current) {

        ListNode* groupStart = current;
        int count = 0;
        while (current && count < k) {
            current = current->next;
            count++;
        }


        if (count == k) {
            ListNode* prev = nullptr;
            ListNode* next = nullptr;
            ListNode* groupEnd = groupStart;


            for (int i = 0; i < k; ++i) {
                next = groupStart->next;
                groupStart->next = prev;
                prev = groupStart;
                groupStart = next;
            }


            prevGroupEnd->next = prev;
            groupEnd->next = current;


            prevGroupEnd = groupEnd;
        }
    }
```

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

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

int main() {
    // Example input
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int k = 3;

    // Create the linked list from the array
    ListNode* head = createList(arr, size);

    // Print the original linked list
    cout << "Original List: ";
    printList(head);

    // Reverse nodes in k-group
    ListNode* modifiedHead = reverseKGroup(head, k);

    // Print the modified linked list
    cout << "Modified List: ";
    printList(modifiedHead);

    return 0;
}
```