

## DAY 4

NAME: - Shailesh Kumar

UID: -22BCS15322

DATE: - 23/12/24

Question 1:-

```
#include <iostream>
```

```
using namespace std;
```

```
int add(int a, int b)
```

```
{
```

```
    return a + b;
```

```
}
```

```
int main()
```

```
{
```

```
    int num1, num2, sum;
```

```
    cout << "Enter two numbers: ";
```

```
    cin >> num1 >> num2;
```

```
    sum = add(num1, num2);
```

```
    cout << "Sum of " << num1 << " and " << num2 << " is: " << sum << endl;
```

```
    return 0;
```

```
}
```

OUTPUT: -

```
Enter two numbers: 4
9
Sum of 4 and 9 is: 13
```

**Question 2: -**

```
#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: ";
```

```
    cin >> n;
```

```
    cout << "Fibonacci Series up to " << n << " terms:" << endl;
```

```
    for (int i = 0; i < n; i++) {
```

```
        cout << fibonacci(i) << " ";
```

```
    }
```

```
    cout << endl;
```

```
    return 0;
```

```
}
```

OUTPUT: -

```
Enter the number of terms: 9
Fibonacci Series up to 9 terms:
0 1 1 2 3 5 8 13 21
```

Question 3: -

```
#include <iostream>
```

```
using namespace std;
```

```
bool isPerfectNumber(int num) {
```

```
    if (num <= 1) {
```

```
        return false;
```

```
    }
```

```
    int sum = 0;
```

```
    for (int i = 1; i <= num / 2; ++i) {
```

```
        if (num % i == 0) {
```

```
            sum += i;
```

```
        }
```

```
    }
```

```
    return sum == num;
```

```
}
```

```
int main() {
```

```
    int number;
```

```

cout << "Enter a number to check if it's a perfect number: ";
cin >> number;

if (isPerfectNumber(number)) {
    cout << number << " is a perfect number." << endl;
} else {
    cout << number << " is not a perfect number." << endl;
}
return 0;
}

```

OUTPUT: -

```

Enter a number to check if it's a perfect number: 28
28 is a perfect number.

```

Question 4: -

```

#include <iostream>
using namespace std;

void addArrays(int arr1[], int arr2[], int result[], int size) {
    for (int i = 0; i < size; ++i) {
        result[i] = arr1[i] + arr2[i];
    }
}

int main() {
    int size = 3;

```

```

int arr1[] = {2, 4, 3};
int arr2[] = {5, 4, 5};
int result[size];

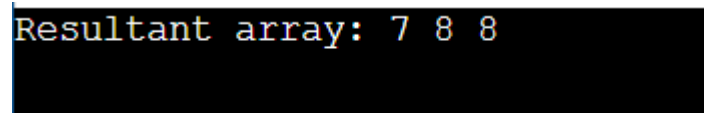
addArrays(arr1, arr2, result, size);

cout << "Resultant array: ";
for (int i = 0; i < size; ++i) {
    cout << result[i] << " ";
}
cout << endl;

return 0;
}

```

OUTPUT: -



```

Resultant array: 7 8 8

```

Question 5: -

```

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

string reverseString(const string& str) {
    string reversed = str;
    int start = 0;

```

```
int end = reversed.length() - 1;

while (start < end) {
    swap(reversed[start], reversed[end]);
    start++;
    end--;
}

return reversed;
}

int main() {
    string input;

    cout << "Enter a string to reverse: ";
    getline(cin, input);

    string reversed = reverseString(input);
    cout << "Reversed string: " << reversed << endl;

    return 0;
}
```

OUTPUT: -

```
Enter a string to reverse: SHIVAM
Reversed string: MAVIHS
```

**Question 6: -**

```
#include <iostream>
```

```
using namespace std;
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
    Node(int val) {
```

```
        data = val;
```

```
        next = nullptr;
```

```
    }
```

```
};
```

```
Node* reverseList(Node* head) {
```

```
    Node* prev = nullptr;
```

```
    Node* current = head;
```

```
    Node* next = nullptr;
```

```
    while (current != nullptr) {
```

```
        next = current->next;
```

```
        current->next = prev;
```

```
        prev = current;
```

```
        current = next;
```

```
    }
```

```
    return prev;
```

```
}
```

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

```
void push(Node*& head, int newData) {  
    Node* newNode = new Node(newData);  
    newNode->next = head;  
    head = newNode;  
}
```

```
int main() {  
    Node* head = nullptr;  
  
    push(head, 3);  
    push(head, 4);  
  
    cout << "Original List: ";  
    printList(head);  
    head = reverseList(head);
```



```
    cout << "Reversed List: ";  
    printList(head);  
  
    return 0;  
}
```

OUTPUT: -

```
Original List: 4 3  
Reversed List: 3 4
```

### Question 7

```
#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 to compute their GCD: ";
    cin >> num1 >> num2;

    int result = gcd(num1, num2);

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

    return 0;
}

```

OUTPUT: -

```

Enter two numbers to compute their GCD: 6
9
The GCD of 6 and 9 is: 3

```

Question 8: -

```

#include <iostream>

#include <cmath>

using namespace std;

bool isPrime(int number) {

    if (number < 2) {
        return false;
    }

    for (int i = 2; i <= sqrt(number); i++) {
        if (number % i == 0) {

```

```

        return false;
    }
}
return true;
}

int main() {
    int num;

    cout << "Enter a number to check if it's prime: ";
    cin >> num;

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

    return 0;
}

```

OUTPUT: -

```

Enter a number to check if it's prime: 45
45 is not a prime number.

```

Question 9: -

```

#include <iostream>

using namespace std;

```

```
void swap(int &a, int &b) {
```

```
    int temp = a;
```

```
    a = b;
```

```
    b = temp;
```

```
}
```

```
int main() {
```

```
    int num1, num2;
```

```
    cout << "Enter two numbers to swap: ";
```

```
    cin >> num1 >> num2;
```

```
    cout << "Before swapping: num1 = " << num1 << ", num2 = " << num2 << endl;
```

```
    swap(num1, num2);
```

```
    cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;
```

```
    return 0;
```

```
}
```

OUTPUT: -

```
Enter two numbers to swap: 8 6
Before swapping: num1 = 8, num2 = 6
After swapping: num1 = 6, num2 = 8
```

**QUESTION 10: -**

```
#include <iostream>
```

```
using namespace std;
```

```
bool isPalindrome(int num) {
```

```
    int originalNum = num;
```

```
    int reversedNum = 0;
```

```
    int remainder;
```

```
    while (num != 0) {
```

```
        remainder = num % 10;
```

```
        reversedNum = reversedNum * 10 + remainder;
```

```
        num /= 10;
```

```
    }
```

```
    return originalNum == reversedNum;
```

```
}
```

```
int main() {
```

```
    int number;
```

```
    cout << "Enter a number: ";
```

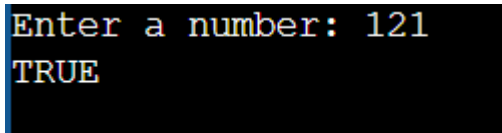
```
    cin >> number;
```

```
    if (isPalindrome(number)) {
```

```
        cout<< "TRUE" << endl;
    } else {
        cout << "FLASE" << endl;
    }

    return 0;
}
```

OUTPUT:-

A terminal window with a black background and a blue border on the left. It displays the text "Enter a number: 121" on the first line and "TRUE" on the second line, both in a light blue monospaced font.

```
Enter a number: 121
TRUE
```

QUESTION 11: -

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

double add(double num1, double num2)
{
    return num1 + num2;
}

double subtract(double num1, double num2) {
    return num1 - num2;
}

double multiply(double num1, double num2) {
    return num1 * num2;
}

double divide(double num1, double num2) {
    return num1 / num2;
}
```

```
}  
  
int main() {  
    double num1, num2, result;  
    int choice;  
    cout << "1. Addition\n";  
    cout << "2. Subtraction\n";  
    cout << "3. Multiplication\n";  
    cout << "4. Division\n";  
    cout << "Enter your choice (1/2/3/4): ";  
    cin >> choice;  
    cout << "Enter first number: ";  
    cin >> num1;  
    cout << "Enter second number: ";  
    cin >> num2;  
    switch (choice) {  
        case 1:  
            result = add(num1, num2);  
            cout << "Result: " << result << endl;  
            break;  
        case 2:  
            result = subtract(num1, num2);  
            cout << "Result: " << result << endl;  
            break;  
        case 3:  
            result = multiply(num1, num2);  
            cout << "Result: " << result << endl;
```

```

        break;
    case 4:
        if (num2 != 0) {
            result = divide(num1, num2);
            cout << "Result: " << result << endl;
        } else {
            cout << "Error! Division by zero is not allowed." << endl;
        }
        break;
    default:
        cout << "Invalid choice! Please select a valid operation." << endl;
}
return 0;
}

```

OUTPUT: -

```

1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice (1/2/3/4): 4
Enter first number: 70/10
Enter second number: Error! Division by zero is not allowed.

```

QUESTION 12: -

```

#include <iostream>

using namespace std;

int sum(int n) {

```



```

    if (n == 1) {
        return 1;
    }

    else {
        return n + sum(n - 1);
    }
}

int main() {
    int n;

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

    int result = sum(n);

    cout << "The sum of the first " << n << " natural numbers is: " << result <<
endl;

    return 0;
}

```

OUTPUT :-

```

Enter a number: 45
The sum of the first 45 natural numbers is: 1035

```

QUESTION 13:-

```
#include <iostream>
```

```
using namespace std;
```

```
int sum(int arr[], int n) {
```

```
    if (n == 0) {
```

```
        return 0;
```

```
    }
```

```
    else {
```

```
        return arr[n-1] + sum(arr, n-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 = sum(arr, n);
```

```
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: 5
6
7
9
8
The sum of the array elements is: 35
```

QUESTION 14: -

```
#include <iostream>
```

```
using namespace std;
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
    Node(int val) : data(val), next(nullptr) {}
```

```
};
```

```
void removeMiddle(Node*& head) {
```

```
    if (head == nullptr || head->next == nullptr) {
```

```
        cout << "List is empty or has only one element, no middle to remove." <<
endl;
```

```
        return;
```

```

    }

    Node *slow = head, *fast = head, *prev = nullptr;

    while (fast != nullptr && fast->next != nullptr) {
        fast = fast->next->next;
        prev = slow;
        slow = slow->next;
    }

    if (prev != nullptr) {
        prev->next = slow->next;
        delete slow;
    }
}

void append(Node*& head, int value) {
    Node* newNode = new Node(value);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

```

```

void printList(Node* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

int main() {
    Node* head = nullptr;
    append(head, 1);
    append(head, 2);
    append(head, 3);
    append(head, 4);
    append(head, 5);
    cout << "Original Linked List: ";
    printList(head);
    removeMiddle(head);
    cout << "Linked List after removing the middle element: ";
    printList(head);
    return 0;
}

```

**OUTPUT: -**

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Linked List after removing the middle element: 1 -> 2 -> 4 -> 5 -> NULL
```

**QUESTION 15: -**

```
#include <iostream>
```

```
#include <stack>
```

```
using namespace std;
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
    Node(int val) : data(val), next(nullptr) {}
```

```
};
```

```
bool isPalindrome(Node* head) {
```

```
    if (head == nullptr || head->next == nullptr) {
```

```
        return true;
```

```
    }
```

```
    stack<int> s;
```

```
    Node* temp = head;
```

```
    while (temp != nullptr) {
```

```
        s.push(temp->data);
```

```
        temp = temp->next;
```

```
    }
```

```
    temp = head;
```

```

while (temp != nullptr) {
    int top = s.top();
    s.pop();
    if (top != temp->data) {
        return false;
    }
    temp = temp->next;
}
return true;
}

void append(Node*& head, int value) {
    Node* newNode = new Node(value);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void printList(Node* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
}

```

```

    }
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

int main() {
    Node* head = nullptr;
    append(head, 1);
    append(head, 2);
    append(head, 3);
    append(head, 2);
    append(head, 1);
    cout << "Linked List: ";
    printList(head);
    if (isPalindrome(head)) {
        cout << "The linked list is a palindrome." << endl;
    } else {
        cout << "The linked list is not a palindrome." << endl;
    }
    return 0;
}

```

**OUTPUT: -**



```
Linked List: 1 -> 2 -> 3 -> 2 -> 1 -> NULL
The linked list is a palindrome.
```

QUESTION 16: -

```
#include <iostream>
```

```
using namespace std;
```

```
int josephus(int n, int k) {
```

```
    if (n == 1) {
```

```
        return 0;
```

```
    }
```

```
    return (josephus(n - 1, k) + k) % n;
```

```
}
```

```
int main() {
```

```
    int n, k;
```

```
    cout << "Enter the number of people: ";
```

```
    cin >> n;
```

```
    cout << "Enter the step count (k): ";
```

```
    cin >> k;
```

```
    int winner = josephus(n, k);
```

```
    cout << "The winner is at position: " << winner + 1 << endl;
```

```
    return 0;
```

```
}
```

OUTPUT: -

```
Enter the number of people: 10
Enter the step count (k): 15
The winner is at position: 6
```

**QUESTION 17: -**

```
#include <iostream>

using namespace std;

struct Node {

    int data;

    Node* next;


    Node(int val) : data(val), next(nullptr) {}

};

Node* reverseKGroup(Node* head, int k) {

    if (!head || k == 1) return head;

    Node* dummy = new Node(0);

    dummy->next = head;

    Node* groupPrev = dummy;

    Node* groupStart = head;

    while (groupStart) {

        Node* groupEnd = groupStart;

        for (int i = 1; i < k && groupEnd; i++) {

            groupEnd = groupEnd->next;

        }

        if (!groupEnd) break;

        Node* nextGroupStart = groupEnd->next;

        groupEnd->next = nullptr;

        Node* prev = nullptr;
```

```

Node* curr = groupStart;
while (curr) {
    Node* nextNode = curr->next;
    curr->next = prev;
    prev = curr;
    curr = nextNode;
}
groupPrev->next = prev;
groupStart->next = nextGroupStart;
groupPrev = groupStart;
groupStart = nextGroupStart;
}
return dummy->next;
}

void append(Node*& head, int value) {
    Node* newNode = new Node(value);
    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

```

```
void printList(Node* head) {
    if (head == nullptr) {
        cout << "List is empty." << endl;
        return;
    }
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}

int main() {
    Node* head = nullptr;
    append(head, 1);
    append(head, 2);
    append(head, 3);
    append(head, 4);
    append(head, 5);
    append(head, 6);
    append(head, 7);
    append(head, 8);
    cout << "Original Linked List: ";
    printList(head);
    int k = 3;
    head = reverseKGroup(head, k);
}
```

```
cout << "Modified Linked List after reversing groups of " << k << ": ";  
printList(head);  
return 0;  
}
```

#### OUTPUT: -

```
Original Linked List: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> NULL  
Modified Linked List after reversing groups of 3: 3 -> 2 -> 1 -> 6 -> 5 -> 4 -> 7 -> 8 -> NULL
```