Name : Lakshita Section : IOT-620

UID: 22BCS15284 Date: 23/12/2024

# Domain Winning Winter Camp

1) Wap To Check If Number Is Palindrome Using Function

Code:

```
#include <iostream>
using namespace std;
bool isPalindrome(int number) {
    int original = number;
    int reversed = 0;
    while (number > 0) {
        int digit = number % 10;  // Get the last digit
        reversed = reversed * 10 + digit;  // Add it to the reversed number
        number /= 10;  // Remove the last digit
    }
    return (original == reversed);
}

int main() {
    int num;

    cout << "Enter a number: ";
    cin >> num;
    cout << (isPalindrome(num) ? "true" : "false") << endl;

    return 0;
}
```

```
Enter a number: 45654
true
```

2) Wap To Sum Of Natural Number Using Recursion

Code:

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

// Function to calculate the sum of natural numbers using recursion
int sumOfNaturalNumbers(int n) {
    // Base case: if n is 1, return 1
    if (n == 1) {
        return 1;
    }
    // Recursive case: sum n + sum of n-1
    return n + sumOfNaturalNumbers(n - 1);
}

int main() {
    int num;
```

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

    // Make sure the number is positive
    if (num < 1) {
        cout << "Please enter a positive integer." << endl;
    } else {
        int result = sumOfNaturalNumbers(num);
        cout << "Sum of first " << num << " natural numbers is: " << result << endl;
    }

    return 0;
}
```

```
Enter a number: 5
Sum of first 5 natural numbers is: 15
```

3) Wap To Sum Of Array Number Using Recursion

Code:

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

// Function to calculate the sum of array elements using recursion
int sumOfArray(int arr[], int size) {
    // Base case: if the size is 0, return 0 (empty array)
    if (size == 0) {
        return 0;
    }
    // Recursive case: sum of the first element + sum of the rest of the array
    return arr[size - 1] + sumOfArray(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 = sumOfArray(arr, n);
    cout << "Sum of array elements is: " << result << endl;

    return 0;
```

```
}
```

```
Enter the number of elements in the array: 5
Enter the elements of the array: 1 2 3 4 5
Sum of array elements is: 15
```

4) Wap To Find The Winner Of Circular Game

Code:

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

// Function to find the winner using the Josephus problem solution
int josephus(int n, int k) {
    // Base case: if there is only one person, they are the winner (position 0)
    if (n == 1)
        return 0;
    // Recursive case: find the position of the winner for n-1 people, then adjust
    return (josephus(n - 1, k) + k) % n;
}

int main() {
    int n, k;

    cout << "Enter the number of people: ";
    cin >> n;

    cout << "Enter the step (k): ";
    cin >> k;

    // Find the winner's position (0-indexed)
    int winner = josephus(n, k);

    // Convert to 1-indexed and display the result
    cout << "The winner is person number: " << winner + 1 << endl;

    return 0;
}
```

Output:

```
Enter the number of people: 8
Enter the step (k): 4
The winner is person number: 6
```

5) Wap To Remove Linked List Element

Code:

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

struct Node {
    int data;
```

```
    Node* next;
};

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

void removeNode(Node*& head, int value) {
    if (!head) return;
    if (head->data == value) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Node* temp = head;
    while (temp->next && temp->next->data != value) temp = temp->next;
    if (temp->next) {
        Node* toDelete = temp->next;
        temp->next = temp->next->next;
        delete toDelete;
    }
}

void display(Node* head) {
    while (head) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int n, value;

    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter elements: ";
    while (n--) {
        cin >> value;
        append(head, value);
    }

    cout << "Enter value to remove: ";
    cin >> value;
```

```
    removeNode(head, value);

    display(head);
    return 0;
}
```

Output:

```
Enter number of elements: 4
Enter elements: 4 5 6 7
Enter value to remove: 5
4 6 7
```

6) Given the head of linked list. Reverse the nodes of the list, k at a time and then return the modified list

Code:

```
#include <iostream>

using namespace std;

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

// Function to reverse a portion of the linked list
ListNode* reverseKGroup(ListNode* head, int k) {
    if (!head || k <= 1) return head;

    // Check if there are at least k nodes left in the list
    ListNode* curr = head;
    int count = 0;
    while (curr && count < k) {
        curr = curr->next;
        count++;
    }

    if (count < k) return head; // Not enough nodes to reverse

    // Reverse k nodes
    ListNode* prev = nullptr;
    curr = head;
    ListNode* next = nullptr;
    count = 0;
    while (curr && count < k) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
        count++;
    }
```

```
    // Recursive call to process the remaining nodes
    if (next) {
        head->next = reverseKGroup(next, k);
    }

    return prev; // New head of the reversed group
}

// Helper function to create a linked list from an array
ListNode* createList(const int arr[], int size) {
    ListNode* head = nullptr;
    ListNode* tail = nullptr;

    for (int i = 0; i < size; ++i) {
        ListNode* newNode = new ListNode(arr[i]);
        if (!head) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }
    return head;
}

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

int main() {
    // Input: linked list and group size k
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int k = 3;

    ListNode* head = createList(arr, 8);

    cout << "Original list: ";
    printList(head);

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

    cout << "Reversed list in groups of " << k << ": ";
    printList(head);

    return 0;
}
```

Output:

```
Original list: 1 2 3 4 5 6 7 8
Reversed list in groups of 3: 3 2 1 6 5 4 7 8
```

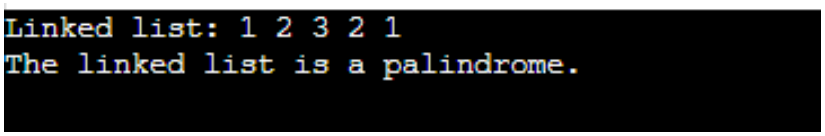
7) Given the head of singly linked list. Return true if the linked list is pallindrome otherwise return false

Code:

```
#include <iostream>
#include <stack>

using namespace std;

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

// Function to check if the linked list is a palindrome
bool isPalindrome(ListNode* head) {
    if (!head || !head->next) return true;

    // Use a slow and fast pointer to find the middle of the linked list
    ListNode* slow = head;
    ListNode* fast = head;
    stack<int> st;

    // Push the first half of the list onto the stack
    while (fast && fast->next) {
        st.push(slow->val);
        slow = slow->next;
        fast = fast->next->next;
    }

    // If the list has an odd number of elements, skip the middle element
    if (fast) {
        slow = slow->next;
    }

    // Compare the second half of the list with the stack
    while (slow) {
        if (st.top() != slow->val) {
            return false; // Not a palindrome
        }
        st.pop();
        slow = slow->next;
    }

    return true; // Palindrome
}
```

```
// Helper function to create a linked list from an array
ListNode* createList(const int arr[], int size) {
    ListNode* head = nullptr;
    ListNode* tail = nullptr;

    for (int i = 0; i < size; ++i) {
        ListNode* newNode = new ListNode(arr[i]);
        if (!head) {
            head = tail = newNode;
        } else {
            tail->next = newNode;
            tail = newNode;
        }
    }
    return head;
}

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

int main() {
    // Input: linked list
    int arr[] = {1, 2, 3, 2, 1};
    ListNode* head = createList(arr, 5);

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

    // Check if the list is a palindrome
    if (isPalindrome(head)) {
        cout << "The linked list is a palindrome." << endl;
    } else {
        cout << "The linked list is not a palindrome." << endl;
    }

    return 0;
}
```

Output:




8) Write a function to check if a number is prime or not

Code:

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

// Function to check if a number is prime
bool isPrime(int num) {
    if (num <= 1) {
        return false; // Numbers less than or equal to 1 are not prime
    }

    // Check divisors from 2 to sqrt(num)
    for (int i = 2; i <= sqrt(num); i++) {
        if (num % i == 0) {
            return false; // Found a divisor, so not prime
        }
    }

    return true; // No divisors found, so the number is prime
}

int main() {
    int number;

    // Input from the user
    cout << "Enter a number to check if it is prime: ";
    cin >> number;

    // Check and display the result
    if (isPrime(number)) {
        cout << number << " is a prime number." << endl;
    } else {
        cout << number << " is not a prime number." << endl;
    }

    return 0;
}
```

```
Enter a number to check if it is prime: 7
7 is a prime number.
```

9) Write a function to add two numbers.

Code:

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

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

int main() {
```

```
    int num1, num2;

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

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

    return 0;
}
```

```
Enter the first number: 5
Enter the second number: 3
The sum of 5 and 3 is: 8
```

10) Write a function to check a number is perfect or not

Code:

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

// Function to check if a number is perfect
bool isPerfectNumber(int num) {
    if (num < 1) {
        return false; // Perfect numbers are positive integers
    }

    int sumOfDivisors = 0;

    // Find the sum of proper divisors
    for (int i = 1; i <= num / 2; i++) {
        if (num % i == 0) {
            sumOfDivisors += i;
        }
    }

    // Check if the sum of divisors equals the number
    return sumOfDivisors == num;
}

int main() {
    int number;

    // Input from the user
    cout << "Enter a number to check if it is perfect: ";
    cin >> number;

    // Check and display the result
    if (isPerfectNumber(number)) {
```

```
        cout << number << " is a perfect number." << endl;
    } else {
        cout << number << " is not a perfect number." << endl;
    }

    return 0;
}
```

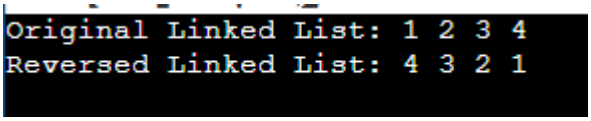
```
Enter a number to check if it is perfect: 6
6 is a perfect number.
```

11) write a program to reverse a linked list using function

Code:

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

// Node structure
struct Node {
    int data;
    Node* next;
};

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

// Function to reverse the linked list
Node* reverseLinkedList(Node* head) {
    Node* prev = nullptr;
    Node* current = head;
    Node* next = nullptr;

    while (current != nullptr) {
        next = current->next; // Store next node
        current->next = prev; // Reverse the link
        prev = current;       // Move prev to current
        current = next;       // Move current to next
    }

    return prev; // New head of the reversed list
}

// Function to print the linked list
void printLinkedList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
```

```
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    // Creating a linked list
    Node* head = createNode(1);
    head->next = createNode(2);
    head->next->next = createNode(3);
    head->next->next->next = createNode(4);

    // Print the original list
    cout << "Original Linked List: ";
    printLinkedList(head);

    // Reverse the linked list
    head = reverseLinkedList(head);

    // Print the reversed list
    cout << "Reversed Linked List: ";
    printLinkedList(head);

    return 0;
}
```




12) Write a recursive function to compute the GCD of two numbers

Code:

```
#include <iostream>

// Function to compute the GCD of two numbers using recursion
int gcd(int a, int b) {
    // Base case: if b is 0, gcd is a
    if (b == 0) {
        return a;
    }
    // Recursive step: gcd(a, b) = gcd(b, a % b)
    return gcd(b, a % b);
}

int main() {
    int num1, num2;
    std::cout << "Enter two numbers: ";
    std::cin >> num1 >> num2;

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

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

```
Enter two numbers: 3150
5520
GCD of 3150 and 5520 is: 30


...Program finished with exit code 0
Press ENTER to exit console.
```

```
    return 0;
```

Output:

13) implement a function that swap two variables using pass by reference in C++

```
#include <iostream>

// Function to swap two variables using pass by reference
void swap(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x, y;
    std::cout << "Enter two numbers: ";
    std::cin >> x >> y;

    std::cout << "Before swapping: x = " << x << ", y = " << y << std::endl;

    swap(x, y); // Call the function to swap

    std::cout << "After swapping: x = " << x << ", y = " << y << std::endl;

    return 0;
}
```

Output:

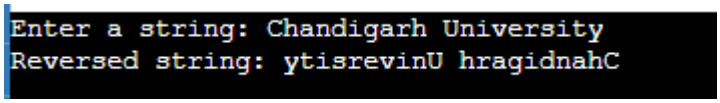
```
Enter two numbers: 20
30
Before swapping: x = 20, y = 30
After swapping: x = 30, y = 20
```

14) Write a program to reverse a string in C++

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

```
// Function to reverse a string
string reverseString(const string &str) {
    string reversed = str;
    int n = reversed.length();

    // Swap characters from start to end
    for (int i = 0; i < n / 2; ++i) {
        swap(reversed[i], reversed[n - i - 1]);
    }
    return reversed;
}

int main() {
    string input;
    cout << "Enter a string: ";
    getline(cin, input); // To read a string with spaces

    string reversed = reverseString(input);

    cout << "Reversed string: " << reversed << endl;

    return 0;
}
```

Output:

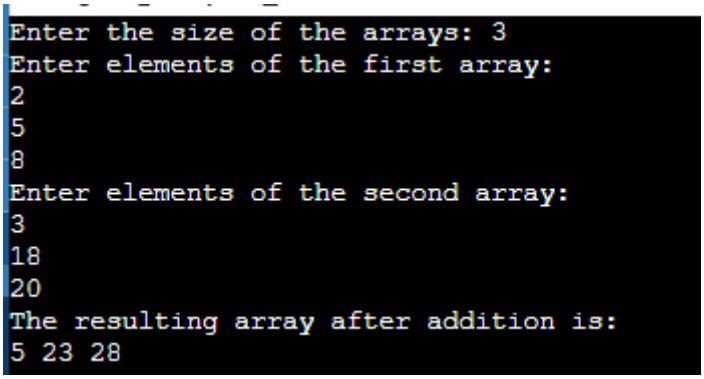



15) Write a program in C++ to add two arrays

```
#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;

    cout << "Enter the size of the arrays: ";
    cin >> size;

    int arr1[size], arr2[size], result[size];

    cout << "Enter elements of the first array:\n";
    for (int i = 0; i < size; ++i) {
        cin >> arr1[i];
```

```
    }

    cout << "Enter elements of the second array:\n";
    for (int i = 0; i < size; ++i) {
        cin >> arr2[i];
    }

    addArrays(arr1, arr2, result, size);

    cout << "The resulting array after addition is:\n";
    for (int i = 0; i < size; ++i) {
        cout << result[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Output:




16) Write A Function To Perform arithmatic Operations Like ( +, -, *, / )

Code:

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

int main() {
    double num1, num2;
    char operation;

    cout << "Enter first number: ";
    cin >> num1;

    cout << "Enter an operator (+, -, *, /): ";
    cin >> operation;

    cout << "Enter second number: ";
    cin >> num2;

    switch (operation) {
        case '+':
```

```
            cout << "Result: " << num1 + num2 << endl;
            break;
        case '-':
            cout << "Result: " << num1 - num2 << endl;
            break;
        case '*':
            cout << "Result: " << num1 * num2 << endl;
            break;
        case '/':
            if (num2 != 0) {
                cout << "Result: " << num1 / num2 << endl;
            } else {
                cout << "Error: Division by zero is not allowed." << endl;
            }
            break;
        default:
            cout << "Error: Invalid operator." << endl;
    }

    return 0;
}
```

```
Enter first number: 5
Enter an operator (+, -, *, /): +
Enter second number: 3
Result: 8
```