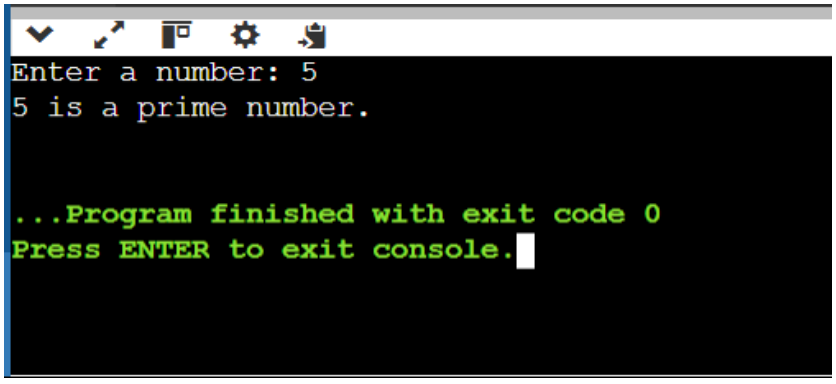
Name: Akshit
UID: 22BCS15486
Section: 620-B

Ques : Number is prime or not

```
#include <iostream>
using namespace std;
bool isPrime(int num) {
    if (num <= 1) return false;
    if (num <= 3) return true;
    if (num % 2 == 0 || num % 3 == 0) return false;
    for (int i = 5; i * i <= num; i += 6) {
        if (num % i == 0 || num % (i + 2) == 0)
            return false;
    }
    return true;
}
int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;
        if (isPrime(number)) {
        cout << number << " is a prime number." << endl;
    } else {
        cout << number << " is not a prime number." << endl;
    }

    return 0;
}
```

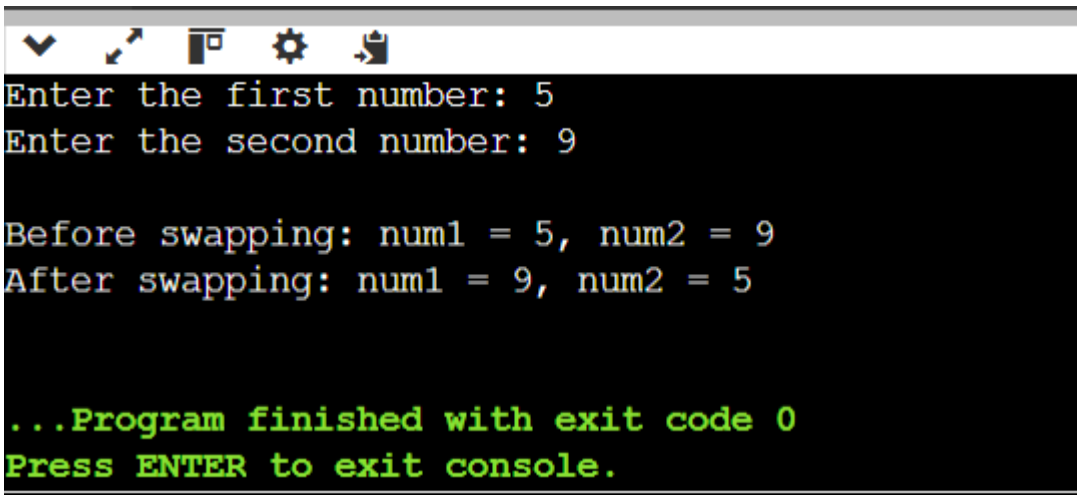
Output:



Ques Swap two numbers

```
#include <iostream>
using namespace std;
void swapNumbers(int &a, int &b) {
   int temp = a;
   a = b;
   b = temp;
}
int main() {
   int num1, num2;
   cout << "Enter the first number: ";
   cin >> num1;
   cout << "Enter the second number: ";
   cin >> num2;
   cout << "\nBefore swapping: num1 = " << num1 << ", num2 = " << num2 << endl;
   swapNumbers(num1, num2);
   cout << "After swapping: num1 = " << num1 << ", num2 = " << num2 << endl;
   return 0;
}
```

Output:

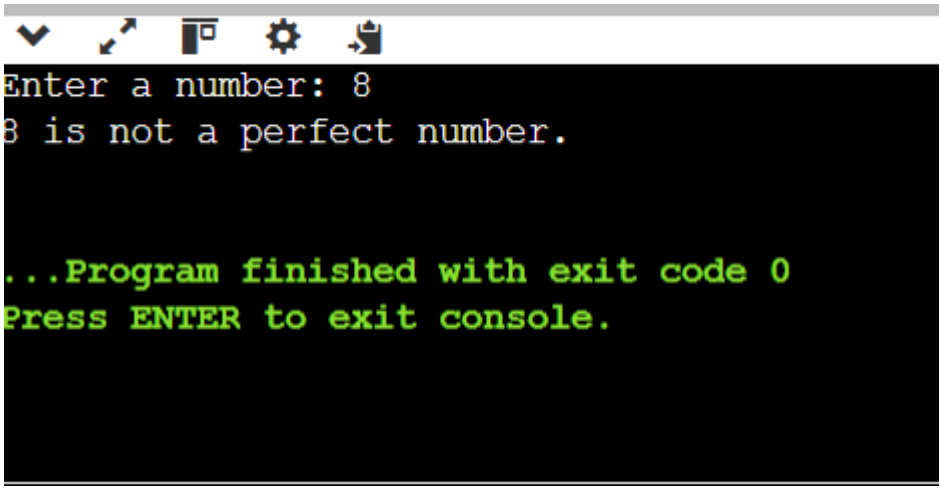


Ques : Number is perfect or not

```
#include <iostream>
using namespace std;
bool isPerfectNumber(int num) {
   if (num <= 0)
      return false;
   int sum = 0;
```

```
    for (int i = 1; i < num; i++) {
        if (num % i == 0) {
            sum += i;
        }
    }
    return sum == num;
}
int main() {
    int number;
    cout << "Enter a 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:



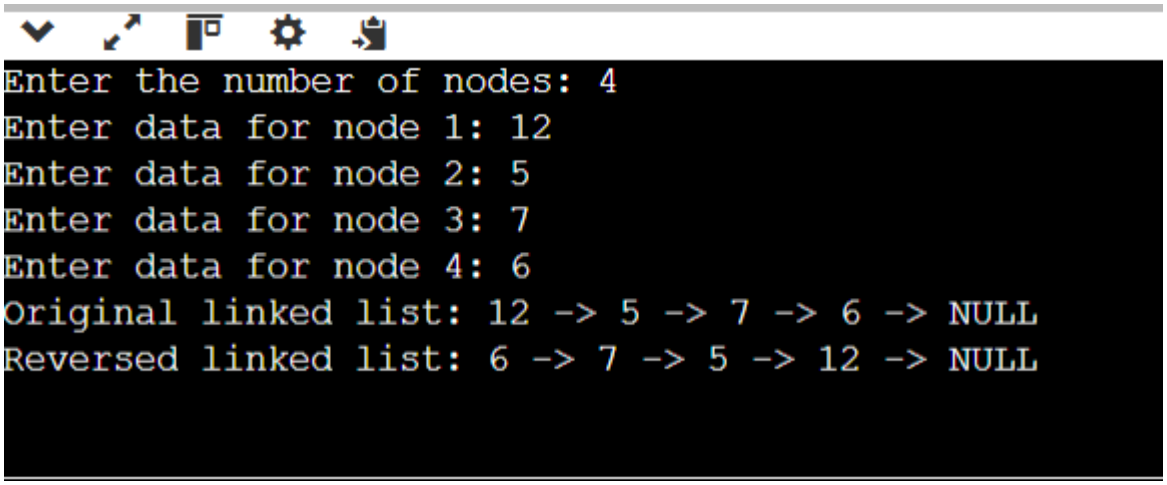
Ques : Reverse a single linked list

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

struct Node {
    int data;
    Node* next;
};
```

```
Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

void insertNode(Node*& head, int data) {
    Node* newNode = createNode(data);
    if (!head) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
}

Node* reverseList(Node* head) {
    Node* prev = nullptr;
    Node* curr = head;
    Node* next = nullptr;

    while (curr) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
void displayList(Node* head) {
    Node* temp = head;
    while (temp) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
```

```
}

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

    cout << "Enter the number of nodes: ";
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cout << "Enter data for node " << i + 1 << ": ";
        cin >> data;
        insertNode(head, data);
    }

    cout << "Original linked list: ";
    displayList(head);

    head = reverseList(head);

    cout << "Reversed linked list: ";
    displayList(head);

    Node* current = head;
    Node* next = nullptr;
    while (current) {
        next = current->next;
        delete current;
        current = next;
    }

    return 0;
}
```

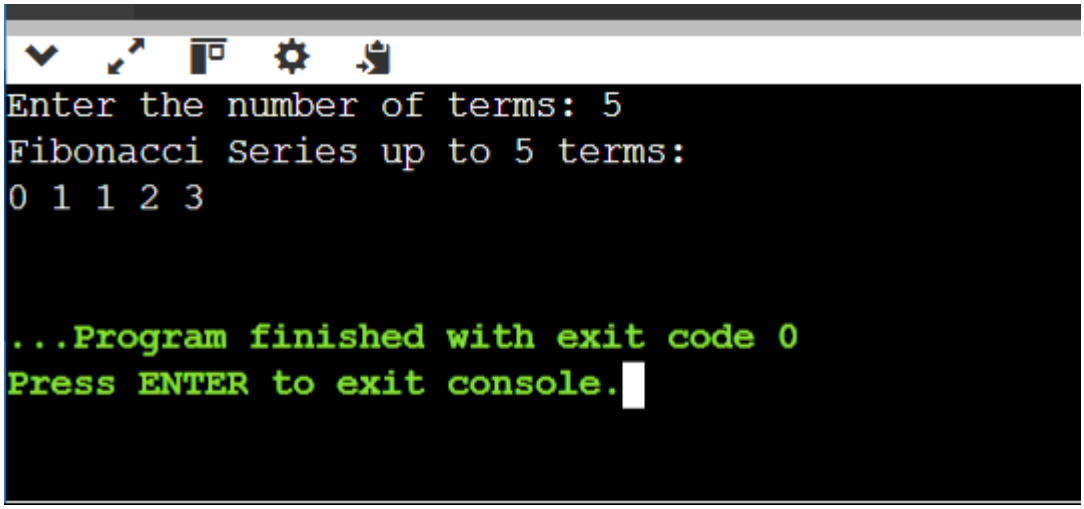
Output:



Ques : Fabonacci series

```
#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:

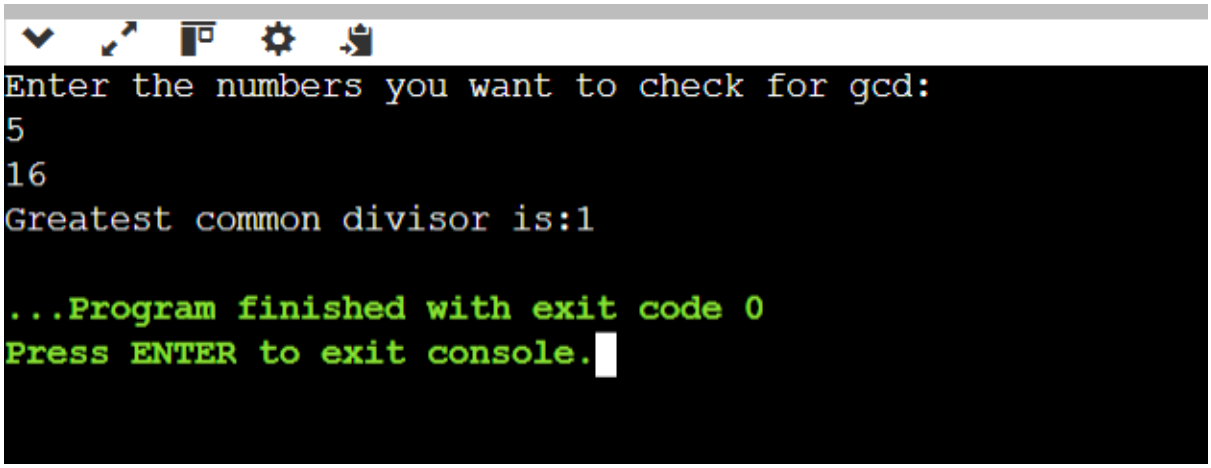


Ques : Writer recursive function to compute the GCD of 2 numbers

```
#include<iostream>
using namespace std;
int main()
{
   int n ,m,max1,max2;
   cout<<"Enter the numbers you want to check for gcd:"<<endl;
   cin>>m>>n;
   if(m>n)
   {
   for(int i=1;i<=n;i++)
   {
      if(m%i==0 && n%i==0)
      {
         max1=i;
      }
      else
      {
         continue;
      }
   }
   cout<<"Greatest common divisor is:"<<max1;
   }
   else
   {
      for(int i=1;i<=m;i++)
   {
      if(m%i==0 && n%i==0)
```

```
        {
            max2=i;
        }
        else
        {
            continue;
        }
    }
    cout<<"Greatest common divisor is:"<<max2;
    }
}
```

Output:



Ques : Add two numbers

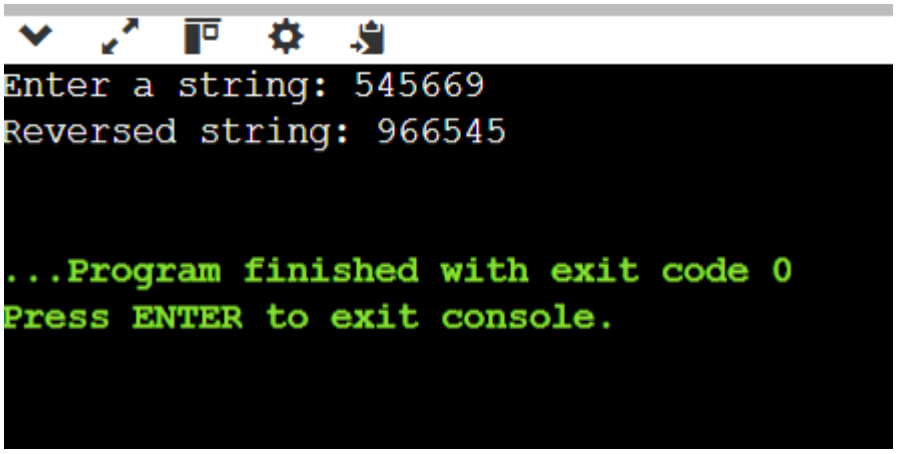
```
#include <iostream>
using namespace std;

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;
}
```

Ques : Write a function to reverse the string

```
#include <iostream>
#include <string>
using namespace std;
string reverseString(string str) {
    int n = str.length();
    for (int i = 0; i < n / 2; ++i) {
        swap(str[i], str[n - i - 1]);
    }
    return str;
}
int main() {
    string input;
    cout << "Enter a string: ";
    cin >> input;
    string reversed = reverseString(input);
    cout << "Reversed string: " << reversed << endl;
    return 0;
}
```

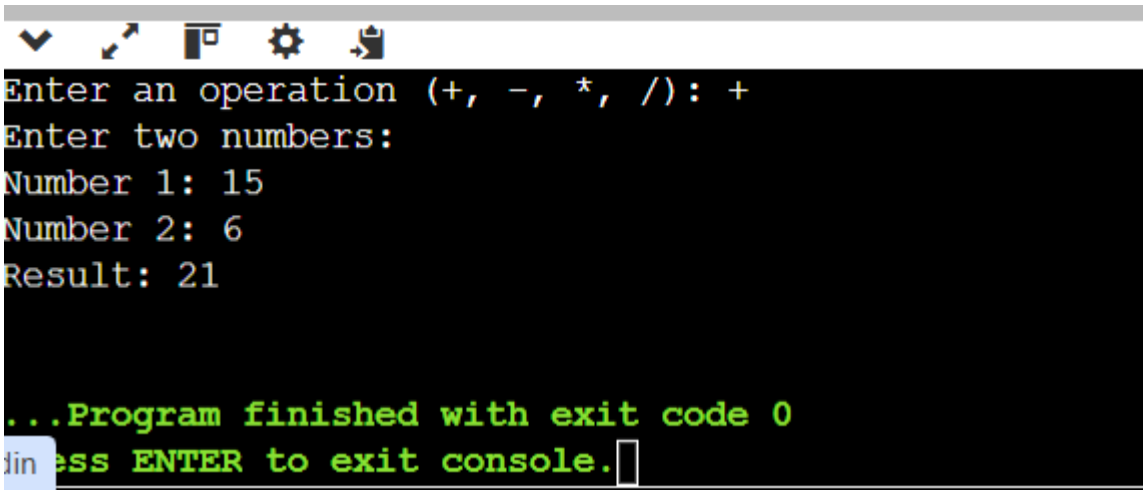
Output:



Ques: write a cpp to create a simple calculator that perform basic operation like addition subtraction multiplication and division

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

int main() {
   char operation;
   double num1, num2;
   cout << "Simple Calculator" << endl;
   cout << "------------------" << endl;
   cout << "Enter an operation (+, -, *, /): ";
   cin >> operation;
   cout << "Enter two numbers: \n";
   cout << "Number 1: ";
   cin >> num1;
   cout << "Number 2: ";
   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 operation." << endl;
   }

   return 0;
}
```

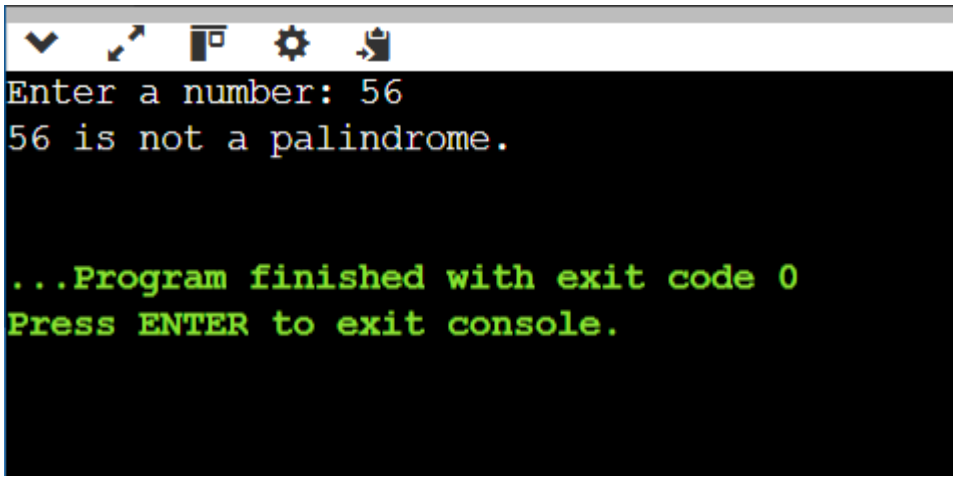
Output:



Ques: write a cpp check if the number is palindrome or not using functions

```
#include <iostream>
using namespace std;
bool isPalindrome(int num) {
    int originalNum = num;
    int reversedNum = 0;

    while (num > 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
        num /= 10;
    }
    return originalNum == reversedNum;
}

int main() {
    int number;
    cout << "Enter a number: ";
    cin >> number;
    if (isPalindrome(number)) {
        cout << number << " is a palindrome." << endl;
    } else {
        cout << number << " is not a palindrome." << endl;
    }

    return 0;
}
```

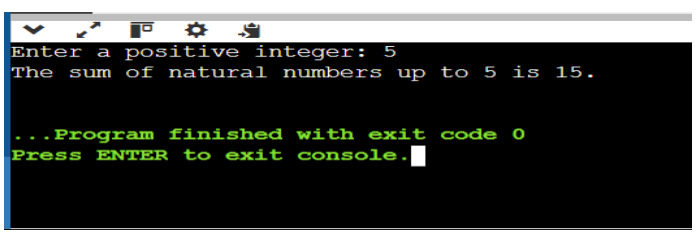
Output:



Ques: sum of natural number using recursion function

```
#include <iostream>
using namespace std;
int sumOfNaturalNumbers(int n) {
    if (n == 0) {
        return 0;
    }
    return n + sumOfNaturalNumbers(n - 1);
}
int main() {
    int number;
    cout << "Enter a positive integer: ";
    cin >> number;
    if (number < 0) {
        cout << "Please enter a positive integer." << endl;
    } else {
        cout << "The sum of natural numbers up to " << number << " is " <<
sumOfNaturalNumbers(number) << "." << endl;
    }
    return 0;
}
```

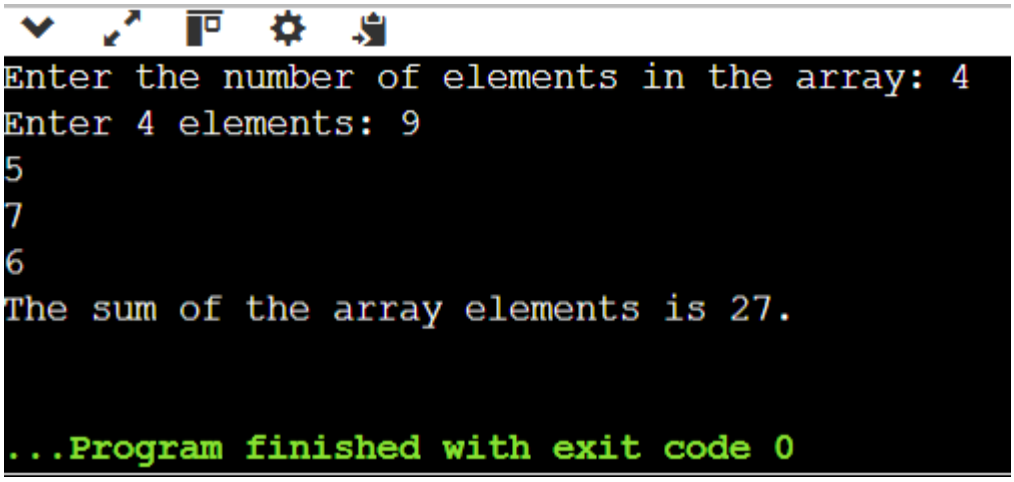
Output:



Ques: sum of array element using recursion function

```
#include <iostream>
using namespace std;
int sumOfArray(int arr[], int size) {
    if (size == 0) {
        return 0;
    }
    return arr[size - 1] + sumOfArray(arr, size - 1);
}
int main() {
    int n;
    cout << "Enter the number of elements in the array: ";
    cin >> n;

    if (n <= 0) {
        cout << "Please enter a positive number of elements." << endl;
        return 1;
    }
    int arr[n];
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    cout << "The sum of the array elements is " << sumOfArray(arr, n) << "." << endl;
    return 0;
}
```

Output:



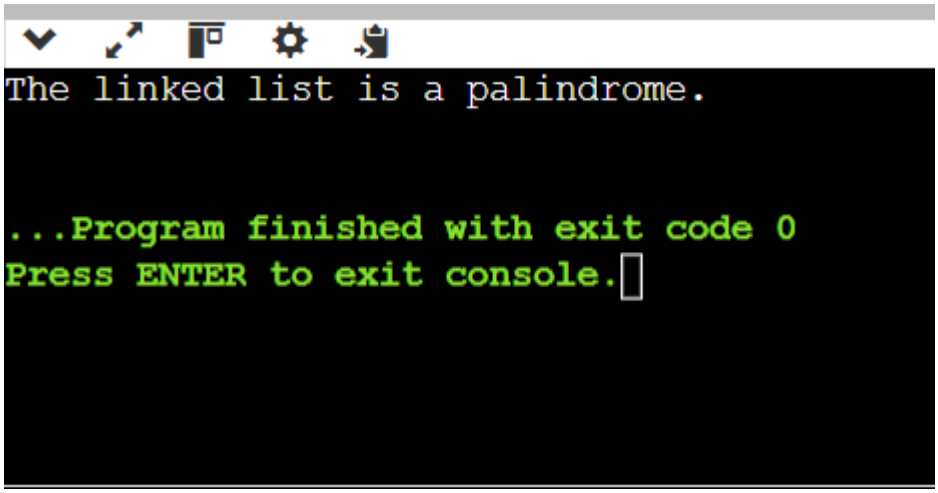
Ques: given the head of linked list is palindrome or not

```
#include <iostream>
```

```
using namespace std;
struct ListNode {
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};
bool isPalindrome(ListNode* head) {
    if (!head || !head->next) {
        return true;
    }
    ListNode* slow = head;
    ListNode* fast = head;
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }
    ListNode* prev = nullptr;
    while (slow) {
        ListNode* temp = slow->next;
        slow->next = prev;
        prev = slow;
        slow = temp;
    }
    ListNode* firstHalf = head;
    ListNode* secondHalf = prev;
    while (secondHalf) {
        if (firstHalf->val != secondHalf->val) {
            return false;
        }
        firstHalf = firstHalf->next;
        secondHalf = secondHalf->next;
    }
    return true;
}
int main() {
    ListNode* head = new ListNode(1);
    head->next = new ListNode(2);
    head->next->next = new ListNode(2);
    head->next->next->next = new ListNode(1);
```

```
    if (isPalindrome(head)) {
        cout << "The linked list is a palindrome." << endl;
    } else {
        cout << "The linked list is not a palindrome." << endl;
    }
    while (head) {
        ListNode* temp = head;
        head = head->next;
        delete temp;
    }

    return 0;
}
```

Output:



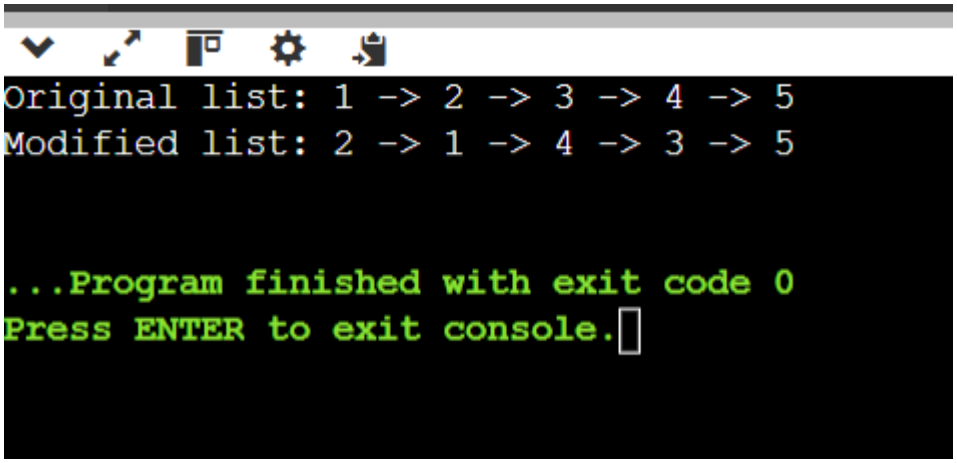
Ques: give the head of the linked list reverse the nodes of list k at a time and return the modified time

```
#include <iostream>
#include <vector>
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(0);
    dummy.next = head;
    ListNode* prevGroupEnd = &dummy;
```

```
    while (true) {
        ListNode* kth = prevGroupEnd;
        for (int i = 0; i < k && kth; ++i) {
            kth = kth->next;
        }
        if (!kth) break;
        ListNode* groupStart = prevGroupEnd->next;
        ListNode* nextGroupStart = kth->next;
        ListNode* prev = nextGroupStart;
        ListNode* curr = groupStart;
        while (curr != nextGroupStart) {
            ListNode* temp = curr->next;
            curr->next = prev;
            prev = curr;
            curr = temp;
        }
        prevGroupEnd->next = kth;
        prevGroupEnd = groupStart;
    }
    return dummy.next;
}
ListNode* createLinkedList(const vector<int>& values) {
    if (values.empty()) return nullptr;
    ListNode* head = new ListNode(values[0]);
    ListNode* current = head;
    for (size_t i = 1; i < values.size(); ++i) {
        current->next = new ListNode(values[i]);
        current = current->next;
    }
    return head;
}
void printLinkedList(ListNode* head) {
    while (head) {
        cout << head->val;
        if (head->next) cout << " -> ";
        head = head->next;
    }
    cout << endl;
}
int main() {
```

```
    vector<int> values = {1, 2, 3, 4, 5};
    int k = 2;

    ListNode* head = createLinkedList(values);
    cout << "Original list: ";
    printLinkedList(head);
    head = reverseKGroup(head, k);
    cout << "Modified list: ";
    printLinkedList(head);
    while (head) {
        ListNode* temp = head;
        head = head->next;
        delete temp;
    }
    return 0;
}
```

Output:



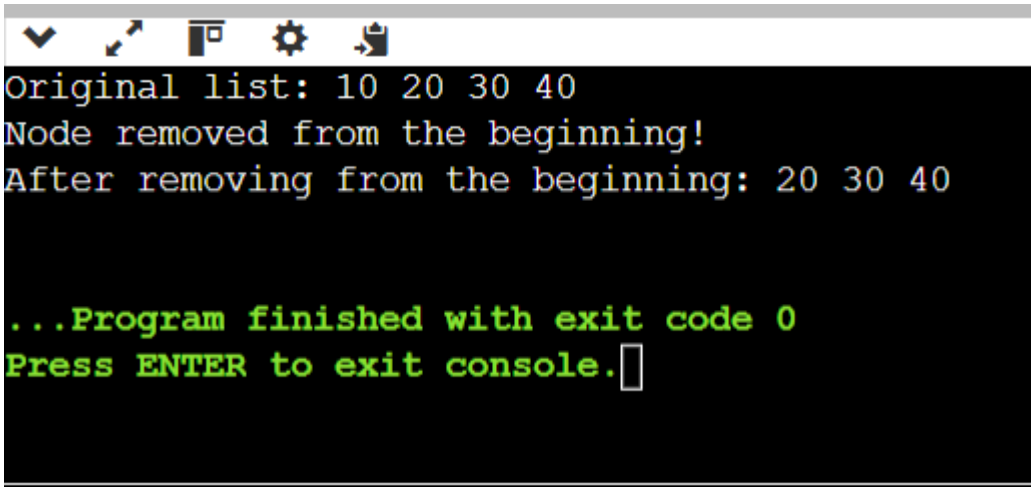
Ques: remove linked list

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
class LinkedList {
private:
    Node* head;
public:
    LinkedList() {
```

```
        head = nullptr;
    }
    void append(int value) {
        Node* newNode = new Node();
        newNode->data = value;
        newNode->next = nullptr;

        if (head == nullptr) {
            head = newNode;
        } else {
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }
    void removeFromBeginning() {
        if (head == nullptr) {
            cout << "List is empty!" << endl;
            return;
        }
        Node* temp = head;
        head = head->next;  // Move the head to the next node
        delete temp;        // Free the old head node
        cout << "Node removed from the beginning!" << endl;
    }
    void display() {
        if (head == nullptr) {
            cout << "List is empty!" << endl;
            return;
        }
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};
```

```
int main() {
    LinkedList list;
    list.append(10);
    list.append(20);
    list.append(30);
    list.append(40);
    cout << "Original list: ";
    list.display();
    list.removeFromBeginning();
    cout << "After removing from the beginning: ";
    list.display();
    return 0;
}
```

Output:



Ques: find the winner of circular game

```
#include <iostream>
#include <vector>
using namespace std;
int findWinner(int n, int k) {
    if (n == 1) {
        return 1;
    }
    return (findWinner(n - 1, k) + k - 1) % n + 1;
}
int main() {
    int n, k;
    cout << "Enter the number of players: ";
    cin >> n;
```

```
    cout << "Enter the step count (k): ";
    cin >> k;
    int winner = findWinner(n, k);
    cout << "The winner of the circular game is player " << winner << "." << endl;
    return 0;
}
```

Output: