1. **Write a program to calculate the Factorial of a number using recursive and non-recursive method**

```cpp
#include<bits/stdc++.h>
using namespace std;
#define ll long long int

ll fact(ll n)
{
    if (n==1)
    return 1;
    return n*fact(n-1);
}
int main()
{
    ll n,c;
    cin >> n;
    cout << "\n1) Recursion\n2)Non-Recursion\nEnter : ";
    cin >> c;
    if (c==1)
    cout << fact(n) << endl;
    else if (c==2)
    {
        ll s=1;
        for (ll i=1;i<=n;i++)
        s*=i;
        cout << s << endl;
    }
    else
    cout << "Invalid\n";

    return 0;
}
```

2. **Write a program to find the nth term F_n of the Fibonacci sequence using recursive and non-recursive method.**

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long int

ll fib(ll n)
{
    if (n == 1 || n == 2)
        return n - 1;
    return fib(n - 1) + fib(n - 2);
}
int main()
{
    ll n, c;
    cin >> n;
    cout << "\n1) Recursion\n2)Non-Recursion\nEnter : ";
    cin >> c;
    if (c == 1)
        cout << fib(n) << endl;
    else if (c == 2)
    {
        if (n == 1)
            cout << "0\n";
        else if (n == 2)
            cout << "1\n";
        else
        {
            ll x = 0, y = 1, i, z;
            for (i = 3; i <= n; i++)
            {
                z = x + y;
                x = y;
                y = z;
            }
            cout << z << endl;
        }
    }
    else
        cout << "Invalid\n";
    return 0;
}
```

3. Write a program to move n disks for Tower of Hanoi problem.

```cpp
#include <bits/stdc++.h>
using namespace std;

void towerOfHanoi(int n, char fr, char tr, char ar)
{
    if (n == 1)
    {
        cout << "Move disk 1 from rod " << fr << " to rod " << tr << endl;
        return;
    }
    towerOfHanoi(n - 1, fr, ar, tr);
    cout << "Move disk " << n << " from rod " << fr << " to rod " << tr << endl;
    towerOfHanoi(n - 1, ar, tr, fr);
}
int main()
{
    int n;
    cout << "Enter the number of disks: ";
    cin >> n;
    towerOfHanoi(n, 'A', 'C', 'B');
    return 0;
}
```

4. Write a program to find the value from Ackerman function.

```cpp
#include <bits/stdc++.h>
using namespace std;
```

```
int ackermann(int m, int n)
{
    if (m == 0)
    {
        return n + 1;
    }
    else if (n == 0)
    {
        return ackermann(m - 1, 1);
    }
    else
    {
        return ackermann(m - 1, ackermann(m, n - 1));
    }
}
int main()
{
    int m, n;
    cin >> m >> n;
    cout <<  ackermann(m, n) << endl;
    return 0;
}
```

5. **Write a program to show the insert and delete operations of a circular queue.**

```
#include <bits/stdc++.h>
using namespace std;
#define n 100
class Queue
{
    int arr[n];
    int f = -1, r = -1, sz = 0;

public:
    void push(int x)
    {
        if (f == -1)
        {
            arr[++r] = x;
            sz++;
            f++;
            return;
        }
```

```cpp
        if (r + 1 == f || (f == 0 && r == n - 1))
        {
            cout << "Overflow\n";
            return;
        }
        if (r == n - 1)
        {
            r = -1;
        }

        arr[++r] = x;
        sz++;
    }
    void pop()
    {
        if (f == -1)
        {
            cout << "Underflow\n";
            return;
        }
        if (f == r)
        {
            f = r = -1;
        }
        if (f == n - 1)
        {
            f = -1;
        }
        f++;
        sz--;
    }
    void front()
    {
        if (f == -1)
        {
            cout << "Underflow\n";
            return;
        }
        cout << "Value : " << arr[f] << endl;
    }
    int Size()
    {
        return sz;
    }
```

```
};
int main()
{
    Queue q;

    while (1)
    {
        cout << "\n1. Push\n2.pop\n3.Front\n4.size\n5.Exit\nChoice : ";
        int x;
        cin >> x;
        if (x == 1)
        {
            int a;
            cout << "Input : ";
            cin >> a;
            q.push(a);
        }
        else if (x == 2)
        {
            q.pop();
        }
        else if (x == 3)
            q.front();
        else if (x == 4)
            cout << "Size : " << q.Size() << endl;
        else if (x == 5 || (x < 0 && x > 5))
            break;
    }
}
```

6. **Write a program to show the insert and delete operations of a priority queue using linked-list.**

```
#include <bits/stdc++.h>
using namespace std;
```

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

    Node(int data, int priority)
    {
        this->data = data;
        this->priority = priority;
        this->next = NULL;
    }
};

class PriorityQueue
{
    Node *head;

public:
    PriorityQueue() : head(NULL) {}

    void push(int data, int priority)
    {
        Node *newNode = new Node(data, priority);

        if (head == NULL || priority > head->priority)
        {
            newNode->next = head;
            head = newNode;
        }
        else
        {
            Node *tmp = head;
            while (tmp->next != NULL && tmp->next->priority > priority)
            {
                tmp = tmp->next;
            }

            newNode->next = tmp->next;
            tmp->next = newNode;
        }
    }

    void pop()
    {
        if (head == NULL)
        {
            cout << "Underflow\n";
            return;
        }

        Node *tmp = head;
        head = head->next;
        delete tmp;
    }

    void display()
    {
        Node *tmp = head;
        while (tmp != NULL)
        {
            cout << "Data: " << tmp->data << ", Priority: " << tmp->priority <<
endl;
            tmp = tmp->next;
        }
    }
};

int main()
{
    PriorityQueue pq;
    pq.push(2, 1);
    pq.push(3, 2);
    pq.push(1, 3);
    pq.display();
    pq.pop();
    pq.display();
    return 0;
}
```

7. Write a program to show the insert and delete operations of a priority queue using array.

```cpp
#include<bits/stdc++.h>
using namespace std;

class PriorityQueue {
public:
    int *arr;
    int size;
    int capacity;

    PriorityQueue(int cap) {
        capacity = cap;
        arr = new int[cap];
        size = 0;
    }

    void insert(int value) {
        if (size == capacity) {
            cout << "Priority Queue is full\n";
            return;
        }
        int i;
        for (i = size - 1; (i >= 0 && arr[i] < value); i--) {
            arr[i + 1] = arr[i];
        }
        arr[i + 1] = value;
        size++;
    }

    void deleteMax() {
        if (size == 0) {
            cout << "Priority Queue is empty\n";
            return;
        }
        size--;
    }

    void display() {
        for (int i = 0; i < size; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};
```

```
int main() {
    PriorityQueue pq(10);
    pq.insert(4);
    pq.insert(1);
    pq.insert(3);
    pq.insert(2);
    pq.insert(16);
    pq.insert(9);
    pq.insert(10);
    pq.display();
    pq.deleteMax();
    pq.display();
    return 0;
}
```

8. Write a program to create a Linked List of n elements and then display the list.

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node(int val) {
        this->val = val;
        this->next = NULL;
    }
};

void insert_at_tail(Node *&head, int v) {
    Node *newNode = new Node(v);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node *tmp = head;
    while (tmp->next != NULL) {
        tmp = tmp->next;
    }
    tmp->next = newNode;
}

void print_linked_list(Node *head) {
    Node *tmp = head;
    while (tmp != NULL) {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

int main() {
    Node *head = NULL;
    int n, value;
    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        insert_at_tail(head, value);
    }
    print_linked_list(head);
    return 0;
}
```

9. Write a program to create a Linked List of n elements and then search an element from the list.

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node(int val) {
        this->val = val;
        this->next = NULL;
    }
};
```

```cpp
void insert_at_tail(Node *&head, int v) {
    Node *newNode = new Node(v);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node *tmp = head;
    while (tmp->next != NULL) {
        tmp = tmp->next;
    }
    tmp->next = newNode;
}

void print_linked_list(Node *head) {
    Node *tmp = head;
    while (tmp != NULL) {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

bool search(Node *head, int key) {
    Node *tmp = head;
    while (tmp != NULL) {
        if (tmp->val == key) {
            return true;
        }
        tmp = tmp->next;
    }
    return false;
}

int main() {
    Node *head = NULL;
    int n, value, key;
    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        insert_at_tail(head, value);
    }
    print_linked_list(head);
    cout << "Enter the element to search: ";
    cin >> key;
    if (search(head, key)) {
        cout << "Element " << key << " found in the list.\n";
    } else {
        cout << "Element " << key << " not found in the list.\n";
    }
    return 0;
}
```

10. Write a program to create a Linked List of n elements and then insert an element to the list.

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node(int val) {
        this->val = val;
        this->next = NULL;
    }
};

void insert_at_tail(Node *&head, int v) {
    Node *newNode = new Node(v);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node *tmp = head;
    while (tmp->next != NULL) {
        tmp = tmp->next;
    }
    tmp->next = newNode;
}

void print_linked_list(Node *head) {
    Node *tmp = head;
    while (tmp != NULL) {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

void insert_at_head(Node *&head, int v) {
    Node *newNode = new Node(v);
    newNode->next = head;
    head = newNode;
}

void insert_at_pos(Node *&head, int pos, int v) {
    if (pos == 1) {
        insert_at_head(head, v);
        return;
    }
    Node *newNode = new Node(v);
    Node *tmp = head;
    for (int i = 1; i < pos - 1; i++) {
        tmp = tmp->next;
        if (tmp == NULL) {
            cout << "Position out of range\n";
            return;
        }
    }
    newNode->next = tmp->next;
    tmp->next = newNode;
}

int main() {
    Node *head = NULL;
    int n, value, pos;
    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        insert_at_tail(head, value);
    }
    print_linked_list(head);
    cout << "Enter the value to insert and position: ";
    cin >> value >> pos;
    insert_at_pos(head, pos, value);
    print_linked_list(head);
    return 0;
}
```

11. Write a program to create a Linked List of n elements and then delete an element from the list.

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node(int val) {
        this->val = val;
        this->next = NULL;
    }
};

void insert_at_tail(Node *&head, int v) {
    Node *newNode = new Node(v);
    if (head == NULL) {
        head = newNode;
        return;
    }
    Node *tmp = head;
    while (tmp->next != NULL) {
        tmp = tmp->next;
    }
    tmp->next = newNode;
}

void print_linked_list(Node *head) {
    Node *tmp = head;
    while (tmp != NULL) {
        cout << tmp->val << " ";
        tmp = tmp->next;
    }
    cout << endl;
```

```cpp
}

void delete_at_head(Node *&head) {
    if (head == NULL) return;
    Node *tmp = head;
    head = head->next;
    delete tmp;
}

void delete_at_pos(Node *&head, int pos) {
    if (pos == 1) {
        delete_at_head(head);
        return;
    }
    Node *tmp = head;
    for (int i = 1; i < pos - 1; i++) {
        tmp = tmp->next;
        if (tmp == NULL || tmp->next == NULL) {
            cout << "Position out of range\n";
            return;
        }
    }
    Node *deleteNode = tmp->next;
    tmp->next = tmp->next->next;
    delete deleteNode;
}

int main() {
    Node *head = NULL;
    int n, value, pos;
    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        insert_at_tail(head, value);
    }
    print_linked_list(head);
    cout << "Enter the position of the element to delete: ";
    cin >> pos;
    delete_at_pos(head, pos);
    print_linked_list(head);
    return 0;
```

```
}
```

12. Write a program to create a Circular Header Linked List of n elements and then display the list.

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node(int val) {
        this->val = val;
        this->next = NULL;
    }
};

class CircularHeaderLinkedList {
public:
    Node *header;

    CircularHeaderLinkedList() {
        header = new Node(0);
        header->next = header;
    }

    void insert(int v) {
        Node *newNode = new Node(v);
        Node *tmp = header;

        while (tmp->next != header) {
            tmp = tmp->next;
        }

        tmp->next = newNode;
        newNode->next = header;
    }
```

```cpp
    void display() {
        if (header->next == header) {
            cout << "List is empty" << endl;
            return;
        }

        Node *tmp = header->next;

        while (tmp != header) {
            cout << tmp->val << " ";
            tmp = tmp->next;
        }

        cout << endl;
    }
};

int main() {
    CircularHeaderLinkedList list;
    int n, value;

    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        list.insert(value);
    }
    list.display();
    return 0;
}
```

13. Write a program to create a Two way Linked List of n elements and then display the list.

```cpp
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int val;
    Node *next;
    Node *prev;
    Node(int val) {
        this->val = val;
        this->next = NULL;
        this->prev = NULL;
    }
};

class DoublyLinkedList {
public:
    Node *head;
    Node *tail;

    DoublyLinkedList() {
        head = NULL;
        tail = NULL;
    }

    void insert_at_tail(int v) {
        Node *newNode = new Node(v);
        if (head == NULL) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }

    void display() {
        Node *tmp = head;
        while (tmp != NULL) {
            cout << tmp->val << " ";
            tmp = tmp->next;
        }
        cout << endl;
    }
};

int main() {
    DoublyLinkedList list;
    int n, value;

    cout << "Enter the number of elements: ";
    cin >> n;
    for (int i = 0; i < n; i++) {
        cout << "Enter value " << i + 1 << ": ";
        cin >> value;
        list.insert_at_tail(value);
    }
    list.display();
    return 0;
}
```

14. Write a program to find the 100!.

```
#include <bits/stdc++.h>
using namespace std;

void multiply(vector<int> &result, int number) {
    int carry = 0;
    for (int i = 0; i < result.size(); i++) {
        int product = result[i] * number + carry;
        result[i] = product % 10;
        carry = product / 10;
    }
    while (carry) {
        result.push_back(carry % 10);
        carry /= 10;
    }
}

void factorial(int n) {
    vector<int> result(1, 1);

    for (int i = 2; i <= n; i++) {
        multiply(result, i);
    }

    // Print the result in reverse order
    for (int i = result.size() - 1; i >= 0; i--) {
        cout << result[i];
    }
    cout << endl;
}

int main() {
    int n = 100;
    cout << "Factorial of " << n << " is: ";
    factorial(n);
    return 0;
}
```

15. Write a program to determine the value of the nth Fibonacci number F n where F n = F n–1 + F n-2 and F 1 = F 2 = 1 and n <= 500.

```
#include <bits/stdc++.h>
using namespace std;

string sumfib(string a, string b)
{
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    while (a.length() < b.length())
    {
        a += '0';
    }
    while (b.length() < a.length())
    {
        b += '0';
    }
    string sum = "";
    int carry = 0;

    for (int i = 0; i < a.length(); i++)
    {
        int s = carry + a[i] - '0' + b[i] - '0';
        int val = s % 10;
        carry = s / 10;
        sum += (char)val + '0';
    }
    if (carry != 0)
        sum += carry + '0';
    reverse(sum.begin(), sum.end());
    return sum;
```

```cpp
}

string fib(int n)
{
    if (n <= 2)
        return "1";
    string a = "1";
    string b = "1", c;
    for (int i = 3; i <= n; i++)
    {
        c = sumfib(a, b);
        a = b;
        b = c;
    }
    return c;
}

int main()
{
    int n;
    cout << "Enter fibonacci number: " << endl;
    cin >> n;
    cout << "Fibonacci of " << n << ": " << fib(n);
    return 0;
}
```