**Question No: 04**

#include<iostream>

using namespace std;

class Node{

public:

int data;

Node\* next;

Node(int d = 0){

data = d;

this->next = NULL;

}

};

class LinkedList{

Node\* head ;

public:

LinkedList(){

head = NULL;

}

void insertAtStart(int d){

Node\* newNode = new Node(d);

newNode->next = head;

head = newNode;

}

void insertAtLast(int d){

Node\* newNode = new Node(d);

Node \*temp = head;

while(temp->next != NULL){

temp = temp->next;

}

temp->next = newNode;

}

void insertAtPosition(int d, int position){

if(position < 1){

cout<<"InValid Position"<<endl;

}

else if(position == 1){

insertAtStart(d);

}

else {

Node\* newNode = new Node(d);

Node\* temp = head;

for( int i =1 ; i<position-1; i++){

if(temp == NULL){

cout<<"InValid Position"<<endl;

break;

}

temp= temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

}

}

void displayLinkedList(){

Node \*temp2 = head;

while(temp2 != NULL){

cout<<temp2->data;

if(temp2->next != NULL){

cout<<",";

}

temp2 = temp2->next;

}

cout<<endl;

}

};

int main(){

LinkedList l;

l.insertAtStart(12);

l.insertAtLast(7);

l.insertAtPosition(24,1);

cout<<"Linked List : ";

l.displayLinkedList();

return 0;

}

**OUTPUT**

****

**Explanation**

In this code implements a singly linked list with methods to insert nodes at the start, end, or a specific position, and displays the list. It handles invalid positions and shows a list with values 24, 12, 7. The structure can be extended to include features like node deletion, searching, or reversing.