**LINKED LIST BASIC:**

#include <bits/stdc++.h>

using namespace std;

struct Node{

int data;

Node\* next;

Node(int x){

data=x;

next=NULL;

}

};

int main()

{

Node \*head=new Node(10);

Node \*temp1=new Node(20);

Node \*temp2=new Node(30);

head->next=temp1;

temp1->next=temp2;

cout<<head->data<<"-->"<<temp1->data<<"-->"<<temp2->data;

return 0;

}

**Linked List Traversal - Longer Method:**

#include <bits/stdc++.h>

using namespace std;

class Node {

public:

int data;

Node\* next;

};

// This function prints contents of linked list

// starting from the given node

void printList(Node\* n)

{

while (n != NULL) {

cout << n->data << " ";

n = n->next;

}

}

// Driver code

int main()

{

Node\* head = NULL;

Node\* second = NULL;

Node\* third = NULL;

// allocate 3 nodes in the heap

head = new Node();

second = new Node();

third = new Node();

head->data = 1; // assign data in first node

head->next = second; // Link first node with second

second->data = 2; // assign data to second node

second->next = third;

third->data = 3; // assign data to third node

third->next = NULL;

printList(head);

return 0;

}

**Linked List Traversal - Easy Method:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=new Node(40);**

**printlist(head);**

**return 0;**

**}**

**Linked List - Insert at Begin:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**Node \*insertBegin(Node \*head,int x){**

**Node \*temp=new Node(x);**

**temp->next=head;**

**return temp;**

**}**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}**

**}**

**int main()**

**{**

**Node \*head=NULL;**

**head=insertBegin(head,30);**

**head=insertBegin(head,20);**

**head=insertBegin(head,10);**

**printlist(head);**

**return 0;**

**}**

**Linked List - Insert at the end:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**Node \*insertEnd(Node \*head,int x){**

**Node \*temp=new Node(x);**

**if(head==NULL)return temp;**

**Node \*curr=head;**

**while(curr->next!=NULL){**

**curr=curr->next;**

**}**

**//temp->next=NULL;**

**curr->next=temp;**

**return head;**

**}**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}**

**}**

**int main()**

**{**

**Node \*head=NULL;**

**head=insertEnd(head,10);**

**head=insertEnd(head,20);**

**head=insertEnd(head,30);**

**printlist(head);**

**return 0;**

**}**

**Linked List To search an element in the iterative method:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**int search(Node \* head, int x){**

**int pos=1;**

**Node \*curr=head;**

**while(curr!=NULL){**

**if(curr->data==x)**

**return pos;**

**else{**

**pos++;**

**curr=curr->next;**

**}**

**}**

**return -1;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**printlist(head);**

**cout<<"Position of element in Linked List: "<<search(head,30);**

**return 0;**

**}**

**Linked List To search an element recursive type:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**int search(Node \* head, int x){**

**if(head==NULL)return -1;**

**if(head->data==x)return 1;**

**else{**

**int res=search(head->next,x);**

**if(res==-1)return -1;**

**else return res+1;**

**}**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**printlist(head);**

**cout<<"Position of element in Linked List: "<<search(head,20);**

**return 0;**

**}**

**Linked List to delete a head node:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*delHead(Node \*head){**

**if(head==NULL)return NULL;**

**else{**

**Node \*temp=head->next;**

**delete(head);**

**return temp;**

**}**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**printlist(head);**

**head=delHead(head);**

**printlist(head);**

**return 0;**

**}**

**Linked List to delete the last node:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int x){**

**data=x;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*delTail(Node \*head){**

**if(head==NULL)return NULL;**

**if(head->next==NULL){**

**delete head;**

**return NULL;**

**}**

**Node \*curr=head;**

**while(curr->next->next!=NULL)**

**curr=curr->next;**

**delete (curr->next);**

**curr->next=NULL;**

**return head;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**printlist(head);**

**head=delTail(head);**

**printlist(head);**

**return 0;**

**}**

**Double Linked List:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* prev;**

**Node\* next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->prev=head;**

**temp1->next=temp2;**

**temp2->prev=temp1;**

**printlist(head);**

**return 0;**

**}**

**DDL-DOUBLE LINKED LIST.**

**DLL insert at begin:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* prev;**

**Node\* next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*insertBegin(Node \*head,int data){**

**Node \*temp=new Node(data);**

**temp->next=head;**

**if(head!=NULL)head->prev=temp;**

**return temp;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->prev=head;**

**temp1->next=temp2;**

**temp2->prev=temp1;**

**head=insertBegin(head,5);**

**printlist(head);**

**return 0;**

**}**

**DLL insert at end:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* prev;**

**Node\* next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*insertEnd(Node \*head,int data){**

**Node \*temp=new Node(data);**

**if(head==NULL)return temp;**

**Node \*curr=head;**

**while(curr->next!=NULL){**

**curr=curr->next;**

**}**

**curr->next=temp;**

**temp->prev=curr;**

**return head;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->prev=head;**

**temp1->next=temp2;**

**temp2->prev=temp1;**

**head=insertEnd(head,40);**

**printlist(head);**

**return 0;**

**}**

**DLL delete head:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* prev;**

**Node\* next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*delHead(Node \*head){**

**if(head==NULL)return NULL;**

**if(head->next==NULL){**

**delete head;**

**return NULL;**

**}**

**else{**

**Node \*temp=head;**

**head=head->next;**

**head->prev=NULL;**

**delete temp;**

**return head;**

**}**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->prev=head;**

**temp1->next=temp2;**

**temp2->prev=temp1;**

**head=delHead(head);**

**printlist(head);**

**return 0;**

**}**

**DLL delete last:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* prev;**

**Node\* next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**Node \*curr=head;**

**while(curr!=NULL){**

**cout<<curr->data<<" ";**

**curr=curr->next;**

**}cout<<endl;**

**}**

**Node \*delLast(Node \*head){**

**if(head==NULL)return NULL;**

**if(head->next==NULL){**

**delete head;**

**return NULL;**

**}**

**Node \*curr=head;**

**while(curr->next!=NULL)**

**curr=curr->next;**

**curr->prev->next=NULL;**

**delete curr;**

**return head;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->prev=head;**

**temp1->next=temp2;**

**temp2->prev=temp1;**

**head=delLast(head);**

**printlist(head);**

**return 0;**

**}**

**CSLL-CIRCULAR SINGLE LINKED LIST:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(5);**

**head->next->next=new Node(20);**

**head->next->next->next=new Node(15);**

**head->next->next->next->next=head;**

**return 0;**

**}**

**CSLL insert at begin:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**if(head==NULL)return;**

**Node \*p=head;**

**do{**

**cout<<p->data<<" ";**

**p=p->next;**

**}while(p!=head);**

**}**

**Node \*insertBegin(Node \* head,int x){**

**Node \*temp=new Node(x);**

**if(head==NULL)**

**temp->next=temp;**

**else{**

**Node \*curr=head;**

**while(curr->next!=head)**

**curr=curr->next;**

**curr->next=temp;**

**temp->next=head;**

**}**

**return temp;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=head;**

**head=insertBegin(head,15);**

**printlist(head);**

**return 0;**

**}**

**CSLL insert at end:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**if(head==NULL)return;**

**Node \*p=head;**

**do{**

**cout<<p->data<<" ";**

**p=p->next;**

**}while(p!=head);**

**}**

**Node \*insertEnd(Node \*head,int x){**

**Node \*temp=new Node(x);**

**if(head==NULL){**

**temp->next=temp;**

**return temp;**

**}**

**else{**

**Node \*curr=head;**

**while(curr->next!=head)**

**curr=curr->next;**

**curr->next=temp;**

**temp->next=head;**

**return head;**

**}**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=head;**

**head=insertEnd(head,15);**

**printlist(head);**

**return 0;**

**}**

**Circular Single Linked List - delete head:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**if(head==NULL)return;**

**Node \*p=head;**

**do{**

**cout<<p->data<<" ";**

**p=p->next;**

**}while(p!=head);**

**}**

**Node \*delHead(Node \*head){**

**if(head==NULL)return NULL;**

**if(head->next==head){**

**delete head;**

**return NULL;**

**}**

**Node \*curr=head;**

**while(curr->next!=head)**

**curr=curr->next;**

**curr->next=head->next;**

**delete head;**

**return (curr->next);**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=new Node(40);**

**head->next->next->next->next=head;**

**head=delHead(head);**

**printlist(head);**

**return 0;**

**}**

**Circular Single Linked List - Delete Kth position:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**if(head==NULL)return;**

**Node \*p=head;**

**do{**

**cout<<p->data<<" ";**

**p=p->next;**

**}while(p!=head);**

**}**

**Node \*deleteHead(Node \*head){**

**if(head==NULL)return NULL;**

**if(head->next==head){**

**delete head;**

**return NULL;**

**}**

**head->data=head->next->data;**

**Node \*temp=head->next;**

**head->next=head->next->next;**

**delete temp;**

**return head;**

**}**

**Node \*deleteKth(Node \*head,int k){**

**if(head==NULL)return head;**

**if(k==1)return deleteHead(head);**

**Node \*curr=head;**

**for(int i=0;i<k-2;i++)**

**curr=curr->next;**

**Node \*temp=curr->next;**

**curr->next=curr->next->next;**

**delete temp;**

**return head;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=new Node(40);**

**head->next->next->next->next=head;**

**head=deleteKth(head,3);**

**printlist(head);**

**return 0;**

**}**

**Circular Double Linked List - Insert at Begin:**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node{**

**int data;**

**Node prev;**

**Node next;**

**Node(int d){**

**data=d;**

**prev=NULL;**

**next=NULL;**

**}**

**};**

**void printlist(Node \*head){**

**if(head==NULL)return;**

**Node \*p=head;**

**do{**

**cout<<p->data<<" ";**

**p=p->next;**

**}while(p!=head);**

**}**

**Node \*insertAtHead(Node \*head,int x){**

**Node \*temp=new Node(x);**

**if(head==NULL){**

**temp->next=temp;**

**temp->prev=temp;**

**return temp;**

**}**

**temp->prev=head->prev;**

**temp->next=head;**

**head->prev->next=temp;**

**head->prev=temp;**

**return temp;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**Node \*temp1=new Node(20);**

**Node \*temp2=new Node(30);**

**head->next=temp1;**

**temp1->next=temp2;**

**temp2->next=head;**

**temp2->prev=temp1;**

**temp1->prev=head;**

**head->prev=temp2;**

**head=insertAtHead(head,5);**

**printlist(head);**

**return 0;**

**}**

**Single Linked List - Insert at any position:**

**#include <bits/stdc++.h> // C++ program for insertion in a single linked**

**// list at a specified position**

**#include <bits/stdc++.h>**

**using namespace std;**

**// A linked list Node**

**struct Node {**

**int data;**

**struct Node\* next;**

**};**

**// Size of linked list**

**int size = 0;**

**// function to create and return a Node**

**Node\* getNode(int data)**

**{**

**// allocating space**

**Node\* newNode = new Node();**

**// inserting the required data**

**newNode->data = data;**

**newNode->next = NULL;**

**return newNode;**

**}**

**// function to insert a Node at required position**

**void insertPos(Node\*\* current, int pos, int data)**

**{**

**// This condition to check whether the**

**// position given is valid or not.**

**if (pos < 1 || pos > size + 1)**

**cout << "Invalid position!" << endl;**

**else {**

**// Keep looping until the pos is zero**

**while (pos--) {**

**if (pos == 0) {**

**// adding Node at required position**

**Node\* temp = getNode(data);**

**// Making the new Node to point to**

**// the old Node at the same position**

**temp->next = current;**

**// Changing the pointer of the Node previous**

**// to the old Node to point to the new Node**

**\*current = temp;**

**}**

**else**

**// Assign double pointer variable to point to the**

**// pointer pointing to the address of next Node**

**current = &(\*current)->next;**

**}**

**size++;**

**}**

**}**

**// This function prints contents**

**// of the linked list**

**void printList(struct Node head)**

**{**

**while (head != NULL) {**

**cout << " " << head->data;**

**head = head->next;**

**}**

**cout << endl;**

**}**

**// Driver Code**

**int main()**

**{**

**// Creating the list 3->5->8->10**

**Node\* head = NULL;**

**head = getNode(3);**

**head->next = getNode(5);**

**head->next->next = getNode(8);**

**head->next->next->next = getNode(10);**

**size = 4;**

**cout << "Linked list before insertion: ";**

**printList(head);**

**int data = 12, pos = 3;**

**insertPos(&head, pos, data);**

**cout << "Linked list after insertion of 12 at position 3: ";**

**printList(head);**

**// front of the linked list**

**data = 1, pos = 1;**

**insertPos(&head, pos, data);**

**cout << "Linked list after insertion of 1 at position 1: ";**

**printList(head);**

**// insertion at end of the linked list**

**data = 15, pos = 7;**

**insertPos(&head, pos, data);**

**cout << "Linked list after insertion of 15 at position 7: ";**

**printList(head);**

**return 0;**

**}**

**insert at any position - easy method:**

**#include <iostream>**

**using namespace std;**

**struct Node{**

**int data;**

**Node\* next;**

**Node(int d){**

**data=d;**

**next=NULL;**

**}**

**};**

**Node \*insertpos(Node \*head, int pos, int data)**

**{**

**Node \*temp=new Node(data);**

**if(pos==1)**

**{**

**temp->next=head;**

**return temp;**

**}**

**Node \*curr=head;**

**for(int i=0;i<pos-2 && curr!=NULL; i++)**

**curr=curr->next;**

**if(curr==NULL)**

**return head;**

**temp->next=curr->next;**

**curr->next=temp;**

**return head;**

**}**

**void printList(Node \*head)**

**{**

**while (head != NULL) {**

**cout << " " << head->data;**

**head = head->next;**

**}**

**cout << endl;**

**}**

**int main()**

**{**

**Node \*head=new Node(10);**

**head->next=new Node(20);**

**head->next->next=new Node(30);**

**head->next->next->next=new Node(40);**

**head->next->next->next->next=NULL;**

**head=insertpos(head,3,5);**

**printList(head);**

**return 0;**

**}**