A purple text on a black background

AI-generated content may be incorrect.

**LAB 8 - Merge two LinkedLists**

Name: Omar Afef Mohammed

Roll number: SU92-BSSEM-S24-086

Section: 3A

Teacher: Mr. Rasikh Ali

Subject: DSA LAB

Date: February 24, 2025

Task:

1. Create 2 Singly LinkedLists and Merge them and display them.

2. Create 2 Double LinkedLists and Merge them and display them.

**ANSWER**

**CODE**

#include <iostream>

using namespace std;

class Snode{

public:

int data;

Snode\* next;

Snode(int d){

data=d;

next=NULL;

}

};

class Slink{

public:

Snode\* head;

Slink(){

head=NULL;

}

void InsertAtStart(int d){

Snode\* newNode= new Snode(d);

newNode->next=head;

head=newNode;

}

void Display() {

Snode\* temp = head;

while (temp != NULL) {

cout << temp->data << " -> ";

temp = temp->next;

}

cout << "NULL\n\n";

}

void Merge(Slink list){

Snode\* temp=this->head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=list.head;

}

};

class Dnode{

public:

int data;

Dnode\* next;

Dnode\* prev;

Dnode(int d){

data=d;

next=NULL;

prev=NULL;

}

};

class Dlink{

public:

Dnode\* head;

Dlink(){

head=NULL;

}

void InsertAtStart(int d){

Dnode\* newNode= new Dnode(d);

newNode->next=head;

if (head != NULL) {

head->prev = newNode;

}

head=newNode;

}

void Display() {

Dnode\* temp = head;

while (temp != NULL) {

cout << temp->data << " <-> ";

temp = temp->next;

}

cout << "NULL\n\n";

}

void Merge(Dlink list){

Dnode\* temp=this->head;

while(temp->next!=NULL){

temp=temp->next;

}

temp->next=list.head;

list.head->prev=temp;

}

};

int main(){

Slink s1,s2;

s1.InsertAtStart(2);

s1.InsertAtStart(1);

s2.InsertAtStart(4);

s2.InsertAtStart(3);

s1.Merge(s2);

s1.Display();

Dlink d1,d2;

d1.InsertAtStart(20);

d1.InsertAtStart(10);

d2.InsertAtStart(40);

d2.InsertAtStart(30);

d1.Merge(d2);

d1.Display();

}

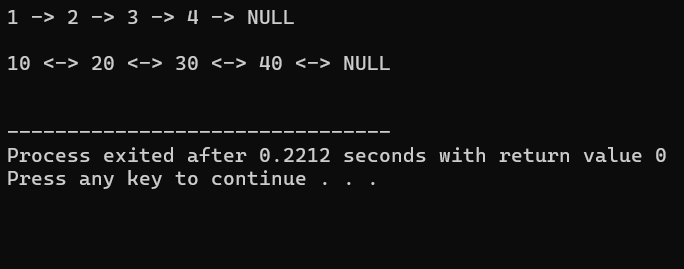
**How it works:**

* **If the first list (this->head) is empty, it is directly assigned to the second list (list.head).**
* **Otherwise, the function traverses to the last node of the first list.**
* **The next pointer of the last node is set to list.head, linking the two lists.**
* **The prev pointer of list.head is updated to point back to the last node of the first list.**

**Why it works:**

* **This allows two doubly and singly linked lists to be combined into one.**
* **Checking for empty lists prevents null pointer errors.**
* **Updating prev ensures correct backward traversal in a doubly linked list.**

**OUTPUT**

****