**Lab-3**

**Lab Work**

1. Fill in the missing code in searchPosOfEle(node first, int x) function which **searches** a given element in the list of elements and prints the **position** of that element if found.

//File Name: SingleLL8.c

//==================================

#include<stdio.h>

#include<stdlib.h>

#include "SearchPositionOfEle.c"

void main() {

NODE first = NULL;

int x, pos, op;

while(1) {

printf("1.Insert At Begin 2.Search an element Position 3.Traverse the List 4.Exit\n");

printf("Enter your option : ");

scanf("%d", &op);

switch(op) {

case 1: printf("Enter an element : ");

scanf("%d", &x);

first = insertAtBegin(first, x);

break;

case 2: printf("Enter search element : ");

scanf("%d", &x);

pos = searchPosOfEle(first, x);

if (pos == 0) {

printf("The given element %d is not found in the given SLL\n", x);

} else {

printf("The given element %d is found at position : %d\n", x, pos);

}

break;

case 3: if (first == NULL) {

printf("Single Linked List is empty\n");

} else {

printf("The elements in SLL are : ");

traverseList(first);

}

break;

case 4: exit(0);

}

}

}

//File Name: SearchPositionOfEle.c

//==================================

struct node {

int data;

struct node \*next;

};

typedef struct node \*NODE;

NODE createNode() {

NODE temp;

temp = (NODE) malloc(sizeof(struct node));

temp -> next = NULL;

return temp;

}

NODE insertAtBegin(NODE first, int x) {

NODE t,p;

p=createNode();

if(first == NULL)

{

first=p;

first->next = NULL;

first->data = x;

}

else

{

t=first;

first = p;

first->data = x;

first->next = t;

}

return first;

}

int searchPosOfEle(NODE first, int key) {

int s=0;

NODE t;

t=first;

while(t != NULL)

{

if(t->data == key)

break;

s++;

t=t->next;

}if(t==NULL)

return 0;

return s+1;

}

void traverseList(NODE first) {

NODE temp = first;

while (temp != NULL)

{

printf("%d --> ",temp -> data);

temp = temp -> next;

}

printf("NULL\n");

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. In the below singly linked list program we have two files, one file contains the **main program** and the other file contains the **functions**, to be implemented by the **user**.  
     
   Here the user has to implement the code for two functions addNodes() and traverseList().  
     
   The addNodes() function creates a new list and adds elements to the list until delimiter -1 is occurred.  
     
   Fill in the missing code in the below functions addNodes(NODE first, int x) and traverseList(NODE first) in the file CreateAndAddNodes.c.

//File Name: SingleLL1.c

//==================================

#include<stdio.h>

#include<stdlib.h>

#include "CreateAndAddNodes.c"

void main() {

NODE first = NULL;

int x;

printf("Enter elements up to -1 : ");

scanf("%d", &x);

while (x != -1) {

first = addNodes(first, x);

scanf("%d", &x);

}

if (first == NULL) {

printf("Single Linked List is empty\n");

} else {

printf("The elements in SLL are : ");

traverseList(first);

}

}

//File Name: CreateAndAddNodes.c

//==================================

struct node {

int data;

struct node \*next;

};

typedef struct node \*NODE;

NODE createNode() {

NODE t;

//struct node\* head;

t=(struct node\*)malloc(sizeof(struct node));

//t->next=data;

t->next = NULL;

return t;

}

NODE addNodes(NODE first, int x) {

NODE p,t;

t=first;

p=createNode();

if(t==NULL)

{

first=p;

first->data=x;

}

else

{

while(t->next!=NULL)

{

t=t->next;

}

t->next=p;

p->data=x;

}

return first;

}

void traverseList(NODE first) {

NODE t;

t=first;

while(t!=NULL)

{

printf("%d --> ",t->data);

t=t->next;

}

printf("NULL\n");

}

**Home Assignment**

1. 1- Fill in the missing code in the below function insertAtPosition(NODE first, int pos, int x) in the file InsAtPosition.c, which inserts a new node at the specified position of the singlly linked list.

//File Name: SingleLL4.c

//==================================

#include<stdio.h>

#include<stdlib.h>

#include "InsAtPosition.c"

void main() {

NODE first = NULL;

int x, pos, op;

while(1) {

printf("1.Insert At specified position 2.Traverse the List 3.Exit\n");

printf("Enter your option : ");

scanf("%d", &op);

switch(op) {

case 1: printf("Enter a position : ");

scanf("%d", &pos);

if (pos <= 0) {

printf("No such position in SLL so insertion is not possible\n");

} else {

printf("Enter an element : ");

scanf("%d", &x);

first = insertAtPosition(first, pos, x);

}

break;

case 2: if (first == NULL) {

printf("Single Linked List is empty\n");

} else {

printf("The elements in SLL are : ");

traverseList(first);

}

break;

case 3: exit(0);

}

}

}

//File Name: InsAtPosition.c

//==================================

struct node {

int data;

struct node \*next;

};

typedef struct node \*NODE;

int count=0;

NODE createNode() {

NODE temp;

temp = (NODE) malloc(sizeof(struct node));

temp -> next = NULL;

return temp;

}

NODE insertAtPosition(NODE first, int pos, int x) {

if(pos-1>count) {printf("No such position in SLL so insertion is not possible\n"); return first;}

NODE temp,cpt,ppt;

temp=createNode();

temp->data=x;

if(first==NULL) {count++; return temp;}

if(pos==1){

temp->next=first;

first=temp;

count++;

return first;

}

cpt=first;

while(--pos){

ppt=cpt;

cpt=cpt->next;

}

ppt->next=temp;

temp->next=cpt;

count++;

return first;

}

void traverseList(NODE first) {

NODE temp;

temp=first;

while(temp!=NULL){

printf("%d --> ",temp->data);

temp=temp->next;

}

printf("NULL\n");

}

1. 2- Fill in the missing code in the below function deleteAtPosition(NODE first, int pos), which deletes a node at the specified position of singly linked list.

//File Name: SingleLL7.c

//==================================

#include <stdio.h>

#include <stdlib.h>

#include "DelAtPosition.c"

void main() {

NODE first = NULL;

int x, op, pos;

while(1) {

printf("1.Insert At End 2.Delete at Position 3.Traverse the List 4.Exit\n");

printf("Enter your option : ");

scanf("%d", &op);

switch(op) {

case 1: printf("Enter an element : ");

scanf("%d", &x);

first = insertAtEnd(first, x);

break;

case 2: if (first == NULL) {

printf("Single Linked List is empty so deletion is not possible\n");

} else {

printf("Enter position : ");

scanf("%d", &pos);

first = deleteAtPosition(first, pos);

}

break;

case 3: if (first == NULL) {

printf("Single Linked List is empty\n");

} else {

printf("The elements in SLL are : ");

traverseList(first);

}

break;

case 4: exit(0);

}

}

}

//File Name: DelAtPosition.c

//==================================

struct node {

int data;

struct node \*next;

};

typedef struct node \*NODE;

NODE createNode() {

NODE temp;

temp = (NODE) malloc(sizeof(struct node));

temp -> next = NULL;

return temp;

}

NODE insertAtEnd(NODE first, int x) {

NODE ptr=createNode();

NODE temp=createNode();

ptr=first;

temp->data=x;

if(first==NULL)

{

first=temp;

return first;

}

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

return first;

}

NODE deleteAtPosition(NODE first, int pos) {

NODE ptr=createNode();

NODE temp=createNode();

int i=1;

ptr=first;

if(pos==1)

{

printf("The deleted element from SLL : %d\n",ptr->data);

first=first->next;

free(ptr);

return first;

}

temp=ptr;

while(ptr!=NULL)

{

if(i==pos){

printf("The deleted element from SLL : %d\n",ptr->data);

temp->next=ptr->next;

free(ptr);

return first;

}

temp=ptr;

ptr=ptr->next;

i++;

}

printf("No such position in SLL so deletion is not possible\n");

return first;

}

void traverseList(NODE first) {

NODE temp = first;

while (temp != NULL) {

printf("%d --> ",temp -> data);

temp = temp -> next;

}

printf("NULL\n");

}

1. 3- Fill in the missing code in the below function void reverseList() in the file ReverseList.c, which reverses the singly linked list.

//File Name: ReverseList.c

//==================================

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

int data;

struct node \*next;

}\*head;

void createList(int n);

void reverseList();

void displayList();

void createList(int n) {

struct node \*temp,\*ptr;

int num ,i;

head=(struct node \*)malloc(sizeof(struct node));

scanf("%d",&num);

head->data=num;

head->next=NULL;

temp=head;

for(i=2;i<=n;i++){

ptr=(struct node \*)malloc(sizeof(struct node));

scanf("%d",&num);

ptr->data=num;

ptr->next=NULL;

temp->next=ptr;

temp=temp->next;

}

}

void reverseList() {

struct node \*prev,\*ptr;

if(head!=NULL)

{

prev=head;

ptr=head->next;

head=head->next;

prev->next=NULL;

while(head!=NULL)

{

head=head->next;

ptr->next=prev;

prev=ptr;

ptr=head;

}

head=prev;

}

}

void displayList() {

struct node \*temp;

if(head == NULL) {

printf("List is empty.");

}

else {

temp = head;

while(temp != NULL) {

printf("%d ", temp->data); // Print the data of current node

temp = temp->next;

}

}

}

int check(int n){

int x;

printf("Enter no.of nodes: ");

scanf("%d", &n);

if (n>0){

while (true){

printf("Enter data: ");

createList(n);

reverseList();

printf("Reversed the list: ");

displayList();

break;}

}

else {

printf("List size must be greater than zero:\n");

return check(n);

}

}

int main(){

int n;

return check(n);

}

1. 5-Fill in the missing code in the below function void insertBeforePosition(int pos, int data, struct Node\*\* head) in the file InsBeforePosition.c, which inserts a new node at the specified position of the singly linked list.

//File Name: SingleLL6.c

//==================================

#include<stdio.h>

#include<stdlib.h>

#include "InsBeforePosition.c"

void main() {

struct Node\* head = NULL;

int n,c,op,pos,x;

while(1){

printf("1. Insert elements in the list.\n2. Insert element before kth index.\n3. Display\n4. Exit\n");

printf("Enter opinion: ");

scanf("%d", &op);

switch(op){

case 1: printf("Enter no.of elements in the list: ");

scanf("%d",&n);

printf("Enter the elements: ");

for(int i=1;i<=n;i++){

scanf("%d",&c);

insertLast(&head,c);

}

break;

case 2: printf("kth index: ");

scanf("%d", &pos);

printf("Element to be inserted before k: ");

scanf("%d",&x);

int size = calcSize(head);

if (pos < 0 || pos >= size) {

printf("No such position in SLL so insertion is not possible\n");

} else {

insertBeforePosition(pos,x,&head);

}

break;

case 3: if (head==NULL){

printf("List is empty\n");

}

else{

display(head);

}

break;

case 4: exit(0);

default: printf("Out of range.\n");

}

}

}

//File Name: InsBeforePosition.c

//==================================

struct Node{

int data;

struct Node \*next;

};

int calcSize(struct Node\* node){

int size=0;

while(node!=NULL){

node = node->next;

size++;

}

return size;

}

void insertLast(struct Node\*\* head, int data){

if(\*head==NULL){

(\*head)=(struct Node \*)malloc(sizeof(struct Node));

(\*head)->data=data;

(\*head)->next=NULL;

}

else{

struct Node \*newnode,\*temp;

temp=(\*head);

while(temp->next!=NULL) temp=temp->next;

newnode=(struct Node \*)malloc(sizeof(struct Node));

newnode->data=data;

newnode->next=NULL;

temp->next=newnode;

}

}

void insertBeforePosition(int pos, int data, struct Node\*\* head) {

if(pos==0){

struct Node \*newnode;

newnode=(struct Node \*)malloc(sizeof(struct Node));

newnode->next=(\*head);

newnode->data=data;

(\*head)=newnode;

}

else{

struct Node \*newnode,\*temp;

temp=(\*head);

while(--pos) temp=temp->next;

newnode=(struct Node \*)malloc(sizeof(struct Node));

newnode->next=temp->next;

newnode->data=data;

temp->next=newnode;

}

}

void display(struct Node\* node){

//as linked list will end when Node is Null

while(node!=NULL){

printf("%d ",node->data);

node = node->next;

}

printf("\n");

}

**Lab-4**

Lab Work

**1-**Write a program that uses functions to perform the following operations on a Circular Linked List:

1. Insert At specified position
2. Traverse the List
3. Search
4. Delete
5. Exit

**2-**Fill in the missing code in the below program to **subtract** two polynomials using linked lists.

//File Name: PolyLLMain2.c

//==================================

#include <stdio.h>

#include <stdlib.h>

#include "SubPolyLL.c"

poly create(poly head) {

poly temp;

char ch;

int coeff, exp;

do {

temp = (poly)malloc(sizeof(struct polynomial));

printf("Enter coeff and exp of node : ");

scanf("%d%d", &coeff, &exp);

temp -> coeff = coeff;

temp -> exp = exp;

temp -> next = NULL;

head = addTerm(head, temp);

printf("Do u want another node (y/n) : ");

scanf(" %c", &ch);

} while(ch != 'n');

return head;

}

void main() {

poly head1 = NULL, head2 = NULL, result = NULL;

int ch;

printf("Enter the first polynomial : \n");

head1 = create(head1);

printf("Enter the second polynomial : \n");

head2 = create(head2);

result = sub(head1, head2);

printf("The first polynomial : ");

print(head1);

printf("The second polynomial : ");

print(head2);

printf("Subtraction result : ");

print(result);

}

//File Name: SubPolyLL.c

//==================================

struct polynomial {

int coeff;

int exp;

struct polynomial \*next;

};

typedef struct polynomial \*poly;

poly createnode(int coeff,int exp)

{

poly temp;

temp = (poly)malloc(sizeof(struct polynomial));

temp->coeff = coeff;

temp->exp = exp;

temp->next = NULL;

return temp;

}

poly addTerm(poly head, poly temp)

{

if(head == NULL)

{

head = temp;

return head;

}

else

{

poly ptr,preptr;

ptr = head;

preptr = ptr;

if(ptr->next == NULL)

{

if(temp->exp > ptr->exp)

{

temp->next = head;

head = temp;

return head;

}

else

{

ptr->next = temp;

return head;

}

}

else

{

int count = 0;

while(temp->exp < ptr->exp)

{

preptr = ptr;

if(ptr->next!=NULL)

{

ptr = ptr->next;

count++;

}

else

break;

}

if(count==0)

{

temp->next = head;

head = temp;

return head;

}

else

{

if(preptr->next == NULL)

{

preptr->next = temp;

return head;

}

else

{

preptr->next = temp;

temp->next = ptr;

return head;

}

}

}

}

}

void print(poly head) {

poly temp = head;

while(temp != NULL) {

printf("%d X^ %d ---> ", temp -> coeff, temp -> exp);

temp = temp -> next;

}

printf("NULL\n");

}

poly sub(poly head1, poly head2)

{

poly result = NULL,ptr1,ptr2,res1 = NULL, temp=NULL;

ptr1 = head1;

ptr2 = head2;

while(ptr1!=NULL && ptr2!=NULL)

{

if(ptr1->exp > ptr2->exp)

{

temp = createnode(ptr1->coeff,ptr1->exp);

ptr1 = ptr1->next;

}

else if(ptr1->exp < ptr2->exp)

{

temp = createnode(ptr2->coeff,ptr2->exp);

ptr2 = ptr2->next;

}

else

{

int diff;

diff = ptr1->coeff - ptr2->coeff;

temp = createnode(diff,ptr1->exp);

ptr1 = ptr1->next;

ptr2 = ptr2->next;

}

if(result==NULL)

{

result = temp;

res1 = result;

}

else

{

res1->next = temp;

res1 = res1->next;

}

}

if(ptr2==NULL)

res1->next = ptr1;

else

{

while(ptr2!=NULL)

{

temp = createnode((-1\*ptr2->coeff),ptr2->exp);

res1->next = temp;

res1 = res1->next;

ptr2 = ptr2->next;

}

}

return result;

}

**Home Assignment**

1- Write a program that uses functions to perform the following **operations on double linked list**

i) Creation  ii)insertion  iii)deletion  iv) Traversal

//File Name: AllOperationsDLL.c

//==================================

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node\*next;

struct node\* prev;

};

typedef struct node\*NODE;

void traverse(NODE head){

NODE temp=head;

while(temp!=NULL){

printf("%d\t",temp->data);

temp=temp->next;

}

printf("\n");

}

NODE deletion(NODE head,int val){

NODE temp=head;

if(head->data==val){

NODE t=head;

head=head->next;

free(t);

}

else{

while(temp->data!=val){

temp=temp->next;

if(temp==NULL)

break;

}

if(temp==NULL){

printf("%d not found.\n",val);

return head;

}

else{

temp->next->prev=temp->prev;

temp->prev->next=temp->next;

free(temp);

}

}

return head;

}

NODE insertattail(NODE head,int val){

NODE p=malloc(sizeof(struct node));

p->data=val;

p->next=NULL;

p->prev=NULL;

if(head==NULL){

head=p;

}

else{

NODE temp=head;

while(temp->next!=NULL){

temp=temp->next;

}

p->prev=temp;

temp->next=p;

}

return head;

}

void main(){

NODE head=NULL;

int ch;

do{

printf("Operations on doubly linked list\n");

printf("1. Insert \n");

printf("2.Remove\n");

printf("3. Display\n");

printf("0. Exit\n");

printf("Enter Choice 0-4? : ");

scanf("%d",&ch);

switch(ch){

case 1:

printf("Enter number: ");

int n;

scanf("%d",&n);

head=insertattail( head , n);

break;

case 2:

printf("Enter number to delete: ");

int k;

scanf("%d",&k);

head = deletion(head,k);

break;

case 3:

traverse(head);

break;

case 0:

break;

}

}

while(ch!=0);

}

1. Fill in the missing code in the below function deleteAtPositionInDCLL(NODE first, int pos), which deletes a node at the specified position of Doubly Circular Linked List.
2. Fill in the missing code in the below program to **add** two polynomials using linked lists.

//File Name: PolyLLMain1.c

//==================================

#include <stdio.h>

#include <stdlib.h>

#include "AddPolyLL.c"

poly create(poly head) {

poly temp;

char ch;

int coeff, exp;

do {

temp = (poly)malloc(sizeof(struct polynomial));

printf("Enter coeff and exp of node : ");

scanf("%d%d", &coeff, &exp);

temp -> coeff = coeff;

temp -> exp = exp;

temp -> next = NULL;

head = addTerm(head, temp);

printf("Do u want another node (y/n) : ");

scanf(" %c", &ch);

} while(ch != 'n');

return head;

}

void main() {

poly head1=NULL, head2= NULL, result = NULL;

int ch;

printf("Enter the first polynomial : \n");

head1 = create(head1);

printf("Enter the second polynomial : \n");

head2 = create(head2);

result = add(head1, head2);

printf("The first polynomial : ");

print(head1);

printf("The second polynomial : ");

print(head2);

printf("Addition result : ");

print(result);

}

//File Name: AddPolyLL.c

//==================================

struct polynomial {

int coeff;

int exp;

struct polynomial \*next;

};

typedef struct polynomial \*poly;

poly addTerm(poly head, poly temp) {

poly ptr1,ptr2;

ptr1 = ptr2=head;

if(ptr1==NULL){

head = temp;

}

else{

while(ptr1!=NULL && ptr1->exp > temp->exp){

ptr2 = ptr1;

ptr1 = ptr1->next;

}

if(ptr1==NULL){

ptr2->next=temp;

}

else if(ptr1->exp == temp->exp){

ptr1->coeff=ptr1->coeff+temp->coeff;

}else if(ptr1->exp < temp ->exp){

if(ptr2 == ptr1){

temp->next=ptr1;

head=temp;

}

else{

temp->next=ptr1;

ptr2->next=temp;

}

}

}

return head;

}

void print(poly head) {

poly temp = head;

while(temp != NULL) {

printf("%d X^ %d ---> ", temp -> coeff, temp -> exp);

temp = temp -> next;

}

printf("NULL\n");

}

poly add(poly head1, poly head2) {

poly p1 =head1;

poly p2 = head2;

poly new;

poly sum;

while(p1!=0 && p2!=0){

new = (poly)malloc(sizeof(struct polynomial ));

new->next = NULL;

if(p1->exp == p2->exp){

new->coeff = p1->coeff+p2->coeff;

new->exp=p1->exp;

p1=p1->next;

p2=p2->next;

}

else if(p1->exp > p2->exp){

new->coeff = p1->coeff;

new->exp = p1->exp;

p1 =p1->next;

}

else if(p1->exp < p2 -> exp){

new->coeff = p2->coeff;

new->exp=p2->exp;

p2=p2->next;

}

sum = addTerm(sum , new);

}

while(p1!=NULL){

new=(poly)malloc(sizeof(struct polynomial));

new->next=NULL;

new->coeff = p1->coeff;

new->exp = p1->exp;

p1=p1->next;

}

while(p2!=NULL){

new = (poly)malloc(sizeof(struct polynomial));

new->next=NULL;

new->coeff = p2->coeff;

new->exp=p2->exp;

p2=p2->next;

}

return sum;

}

1. Fill in the missing code in the below program to **multiply** two polynomials using linked lists.

//File Name: PolyLLMain3.c

//==================================

#include <stdio.h>

#include <stdlib.h>

#include "ProductPolyLL.c"

poly create(poly head) {

poly temp;

char ch;

int coeff, exp;

do {

temp = (poly)malloc(sizeof(struct polynomial));

printf("Enter coeff and exp of node : ");

scanf("%d%d", &coeff, &exp);

temp -> coeff = coeff;

temp -> exp = exp;

temp -> next = NULL;

head = addTerm(head, temp);

printf("Do u want another node (y/n) : ");

scanf(" %c", &ch);

} while(ch != 'n');

return head;

}

void main() {

poly head1 = NULL, head2 = NULL, result = NULL;

int ch;

printf("Enter the first polynomial : \n");

head1 = create(head1);

printf("Enter the second polynomial : \n");

head2 = create(head2);

result = mul(head1, head2);

printf("The first polynomial : ");

print(head1);

printf("The second polynomial : ");

print(head2);

printf("Product result : ");

print(result);

}

//File Name: ProductPolyLL.c

//==================================

struct polynomial {

int coeff;

int exp;

struct polynomial \*next;

};

typedef struct polynomial \*poly;

poly createnode(int coeff,int exp){

poly temp;

temp = (poly)malloc(sizeof(struct polynomial));

temp->coeff = coeff;

temp->exp = exp;

temp->next = NULL;

return temp;

}

poly addTerm(poly head, poly temp)

{

if(head==NULL)

{

head = temp;

return head;

}

else

{

poly ptr,preptr;

ptr= head;

preptr = ptr;

if(ptr->next == NULL)

{

if(temp->exp > ptr ->exp)

{

temp->next = head;

head = temp;

return head;

}

else if(temp->exp < ptr->exp)

{

head->next = temp;

return head;

}

else

{

ptr->coeff = ptr ->coeff + temp->coeff;

return head;

}

}

else

{

int count = 0;

while(temp ->exp < ptr->exp)

{

preptr = ptr;

count++;

if(ptr->next!=NULL)

ptr = ptr->next;

else

break;

}

if(count==0)

{

temp->next = head;

head = temp;

return head;

}

else

{

if(preptr->next ==NULL)

{

preptr->next = temp;

return head;

}

else

{

if(temp->exp == preptr->next->exp)

{

ptr->coeff = ptr ->coeff + temp->coeff;

return head;

}

preptr->next = temp;

temp->next = ptr;

return head;

}

}

}

}

}

void print(poly head) {

poly temp = head;

while(temp != NULL) {

printf("%d X^ %d ---> ", temp -> coeff, temp -> exp);

temp = temp -> next;

}

printf("NULL\n");

}

poly mul(poly head1, poly head2)

{

int coeff,exp;

poly ptr1,ptr2;

poly result = NULL, temp = NULL;

ptr1 = head1;

ptr2 = head2;

while(ptr1!=NULL)

{

ptr2 = head2;

while(ptr2!=NULL)

{

coeff = ptr1->coeff \* ptr2->coeff;

exp = ptr1->exp + ptr2->exp;

temp = createnode(coeff,exp);

result = addTerm(result,temp);

ptr2 = ptr2 ->next;

}

ptr1 = ptr1-> next;

}

return result;

}