1: WRITE A PROGRAM OF DOUBLY LINKED LIST WITH THE FOLLOWING OPERATIONS

* CREATE LIST
* DISPLAY THE LL
* ADD TO EMPTY LSIT
* ADD AT THE BEGINNING
* ADD AT THE END
* ADD AFTER A SPECIFIED NODE
* ADD BEFORE A SPECIFIED NODE
* DELETE

Program:

#include<stdio.h>

#include<stdlib.h>

struct node {

struct node \*prev;

int info;

struct node \*next;

};

struct node \*create\_list(struct node \*start);

void display(struct node \*start);

struct node \*addtoempty(struct node \*start,int data);

struct node \*addatbeg(struct node \*start,int data);

struct node \*addatend(struct node \*start,int data);

struct node \*addafter(struct node \*start,int data,int item);

struct node \*addbefore(struct node \*start,int data,int item);

struct node \*del(struct node \*start,int data);

int main() {

int choice,data,item;

struct node \*start=NULL;

while(1) {

printf("1.Create the List\n");

printf("2.Display the list\n");

printf("3.Add to empty list\n");

printf("4.Add at beginning of the list\n");

printf("5.Add at end of the list\n");

printf("6.Add after\n");

printf("7.Add before\n");

printf("8.Delete\n");

printf("9.Quit\n");

printf("Enter choice : ");

scanf("%d",&choice);

switch(choice) {

case 1:

start=create\_list(start);

break;

case 2:

display(start);

break;

case 3:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

start=addtoempty(start,data);

break;

case 4:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

start=addatbeg(start,data);

break;

case 5:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

start=addatend(start,data);

break;

case 6:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

printf("Enter the element after which to insert : ");

scanf("%d",&item);

start=addafter(start,data,item);

break;

case 7:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

printf("Enter the element before which to insert : ");

scanf("%d",&item);

start=addbefore(start,data,item);

break;

case 8:

printf("Enter the element to be deleted : ");

scanf("%d",&data);

start=del(start,data);

break;

case 9:

exit(1);

default:

printf("Wrong choice\n");

}/\*End of switch\*/

printf("\n\n\n");

}/\*End of while\*/

return 0;

}/\*End of main()\*/

struct node \*create\_list(struct node \*start) {

int i,n,data;

printf("Enter the number of nodes : ");

scanf("%d",&n);

start=NULL;

if(n==0)

return start;

printf("Enter the element to be inserted : ");

scanf("%d",&data);

start=addtoempty(start,data);

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

printf("Enter the element to be inserted : ");

scanf("%d",&data);

start=addatend(start,data);

}

return start;

}/\*End of create\_list()\*/

void display(struct node \*start) {

struct node \*p;

if(start==NULL) {

printf("List is empty\n");

return;

}

p=start;

printf("List is :\n");

while(p!=NULL) {

printf("%d ",p->info);

p=p->next;

}

printf("\n");

}/\*End of display() \*/

struct node \*addtoempty(struct node \*start,int data) {

struct node \*tmp;

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

tmp->info=data;

tmp->prev=NULL;

tmp->next=NULL;

start=tmp;

return start;

}/\*End of addtoempty( )\*/

struct node \*addatbeg(struct node \*start,int data) {

struct node \*tmp;

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

tmp->info=data;

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

return start;

}/\*End of addatbeg( )\*/

struct node \*addatend(struct node \*start,int data) {

struct node \*tmp,\*p;

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

tmp->info=data;

p=start;

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

p->next=tmp;

tmp->next=NULL;

tmp->prev=p;

return start;

}/\*End of addatend( )\*/

struct node \*addafter(struct node \*start,int data,int item) {

struct node \*tmp,\*p;

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

tmp->info=data;

p=start;

while(p!=NULL) {

if(p->info==item) {

tmp->prev=p;

tmp->next=p->next;

if(p->next!=NULL) p->next->prev=tmp;

p->next=tmp;

return start;

}

p=p->next;

}

printf("%d not present in the list\n\n",item);

return start;

}/\*End of addafter()\*/

struct node \*addbefore(struct node \*start,int data,int item) {

struct node \*tmp,\*q;

if(start==NULL ) {

printf("List is empty\n");

return start;

}

if(start->info==item) {

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

tmp->info=data;

tmp->prev=NULL;

tmp->next=start;

start->prev=tmp;

start=tmp;

return start;

}

q=start;

while(q!=NULL) {

if(q->info==item) {

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

tmp->info=data;

tmp->prev=q->prev;

tmp->next = q;

q->prev->next=tmp;

q->prev=tmp;

return start;

}

q=q->next;

}

printf("%d not present in the list\n",item);

return start;

}

struct node \*del(struct node \*start,int data) {

struct node \*tmp;

if(start==NULL) {

printf("List is empty\n");

return start;

}

if(start->next==NULL)

if(start->info==data) {

tmp=start;

start=NULL;

free(tmp);

return start;

}

else

{

printf("Element %d not found\n",data);

return start;

}

if(start->info==data)

{

tmp=start;

start=start->next;

start->prev=NULL;

free(tmp);

return start;

}

/\*Deletion in between\*/

tmp=start->next;

while(tmp->next!=NULL ) {

if(tmp->info==data) {

tmp->prev->next=tmp->next;

tmp->next->prev=tmp->prev;

free(tmp);

return start;

}

tmp=tmp->next;

}

/\*Deletion of last node\*/

if(tmp->info==data) {

tmp->prev->next=NULL;

free(tmp);

return start;

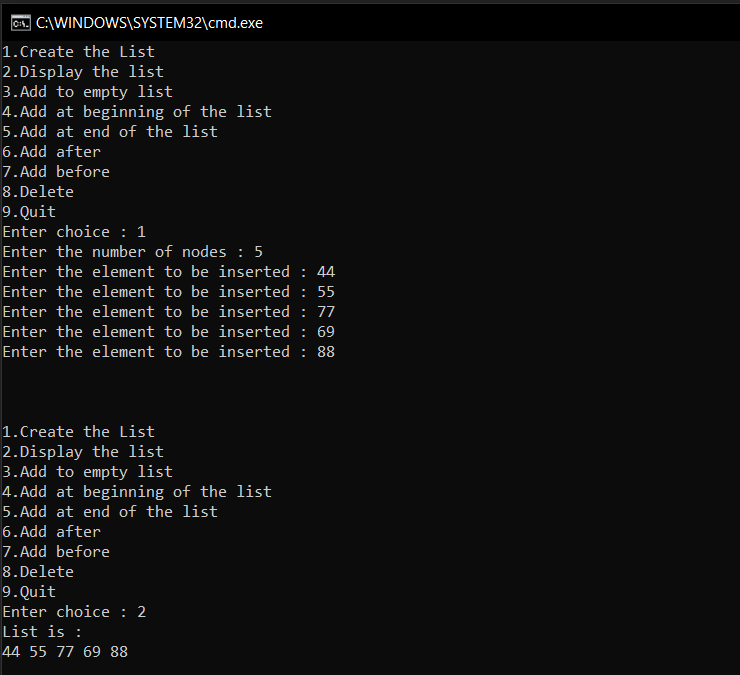
}

printf("Element %d not found\n",data);

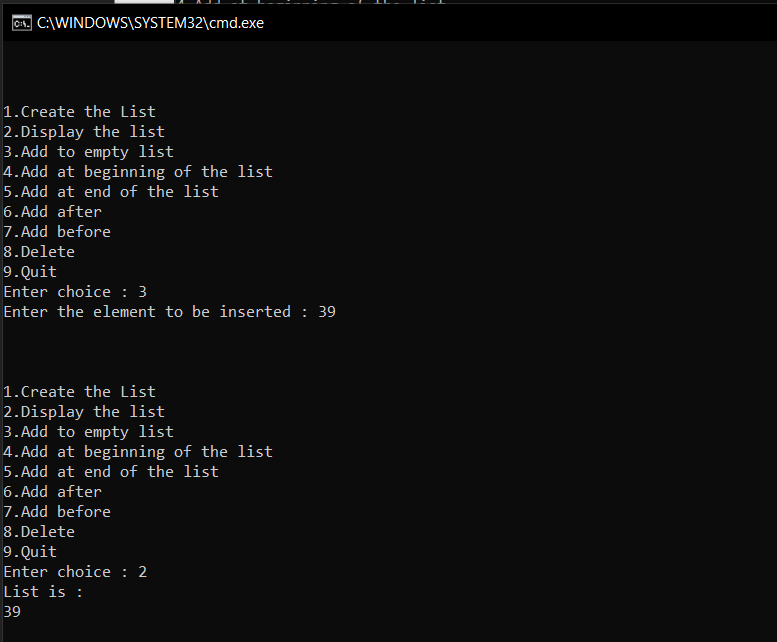
return start;

}

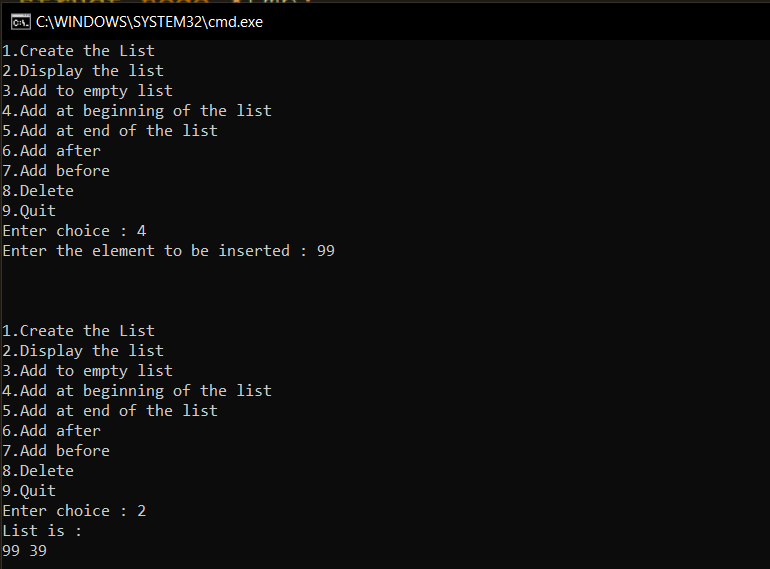
Output 1:



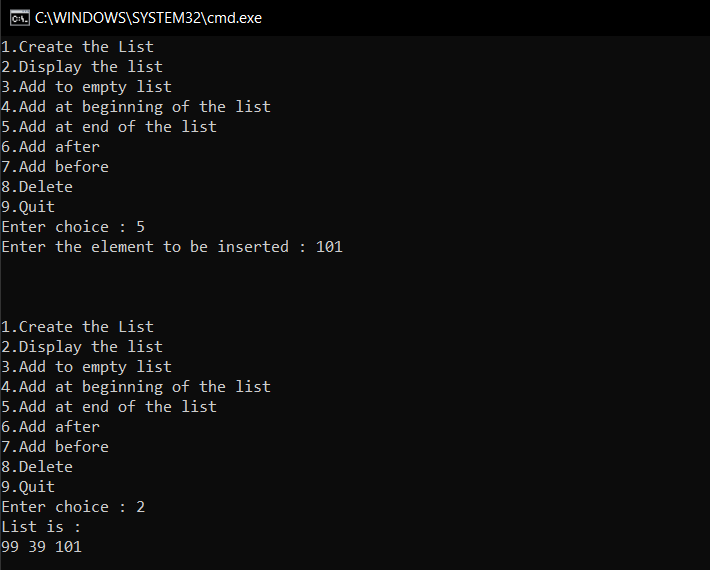
Output 2:



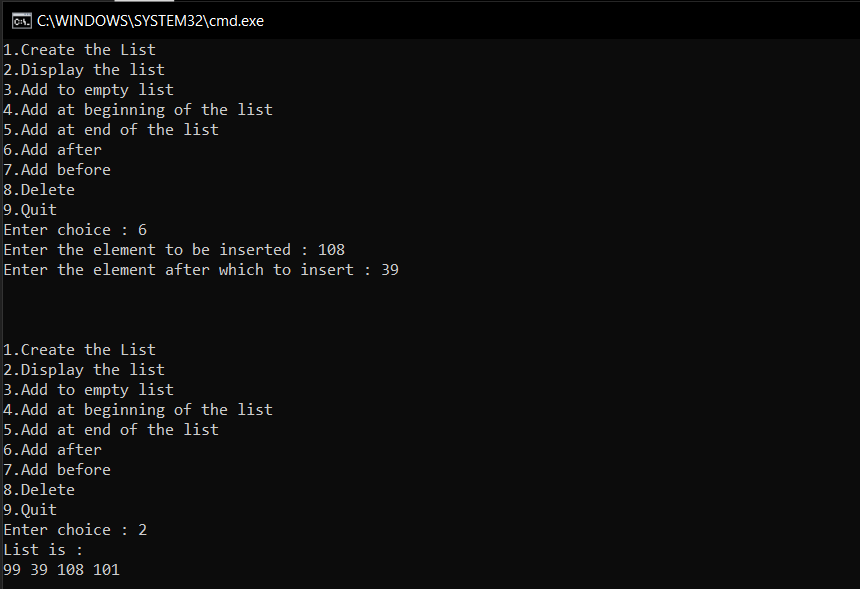
Output 3:



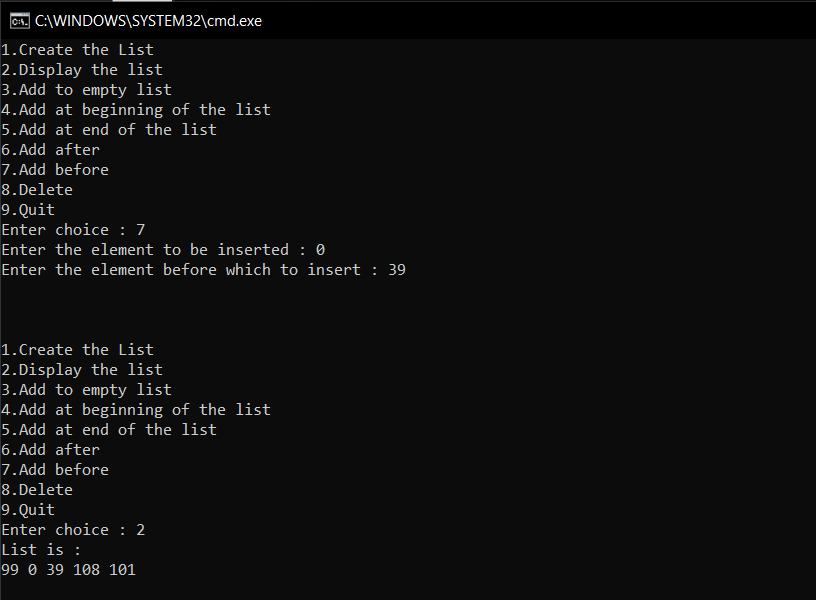
Output 4:



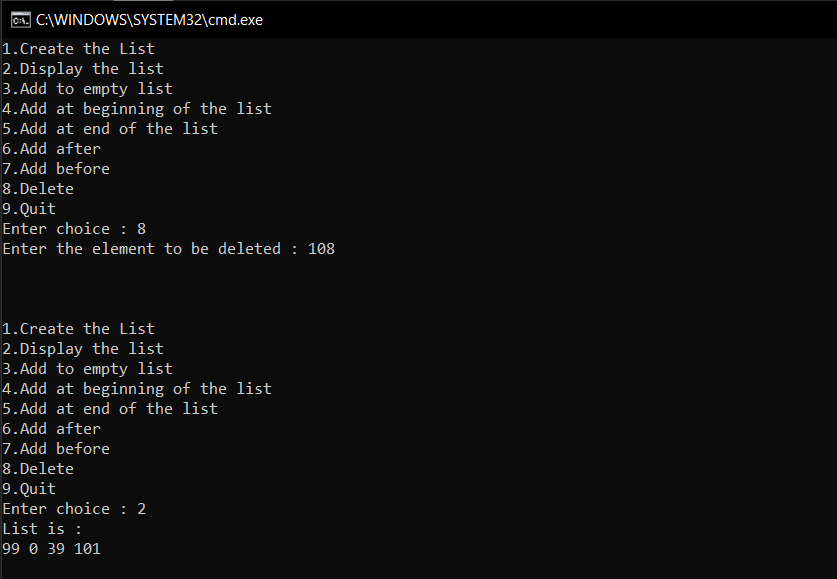
Output 5:



Output 6:



Output 7:



2. Write a program of circular linked list with the following

operations

* Create LL
* Display the LL
* Add to empty list
* Add at the beginning
* Add at the end
* Add after a specified node
* Add before a specified node
* Delete a specified node

Program:

#include<stdio.h>

#include<stdlib.h>

struct node {

int info;

struct node \*link;

};

struct node \*create\_list(struct node \*last);

void display(struct node \*last);

struct node \*addtoempty(struct node \*last,int data);

struct node \*addatbeg(struct node \*last,int data);

struct node \*addatend(struct node \*last,int data);

struct node \*addafter(struct node \*last,int data,int item);

struct node \*addbefore(struct node \*last,int data,int item);

struct node \*del(struct node \*last,int data);

int main( ) {

int choice,data,item;

struct node \*last=NULL;

while(1) {

printf("1.Create the List\n");

printf("2.Display the list\n");

printf("3.Add to empty list\n");

printf("4.Add at beginning\n");

printf("5.Add at end\n");

printf("6.Add after \n");

printf("7.Add before \n");

printf("8.Delete\n");

printf("9.Quit\n");

printf("Enter your choice : ");

scanf("%d",&choice);

switch(choice) {

case 1:

last=create\_list(last);

break;

case 2:

display(last);

break;

case 3:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

last=addtoempty(last,data);

break;

case 4:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

last=addatbeg(last,data);

break;

case 5:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

last=addatend(last,data);

break;

case 6:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

printf("Enter the element after which to insert : ");

scanf("%d",&item);

last=addafter(last,data,item);

break;

case 7:

printf("Enter the element to be inserted : ");

scanf("%d",&data);

printf("Enter the element before which to insert : ");

scanf("%d",&item);

last=addbefore(last,data,item);

break;

case 8:

printf("Enter the element to be deleted : ");

scanf("%d",&data);

last=del(last,data);

break;

case 9:

exit(1);

default:

printf("Wrong choice\n");

}/\*End of switch\*/

printf("\n\n\n");

}/\*End of while\*/

return 0;

}/\*End of main( )\*/

struct node \*create\_list(struct node \*last) {

int i,n,data;

printf("Enter the number of nodes : ");

scanf("%d",&n);

last=NULL;

if(n==0) return last;

printf("Enter the element to be inserted : ");

scanf("%d",&data);

last=addtoempty(last,data);

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

printf("Enter the element to be inserted : ");

scanf("%d",&data);

last=addatend(last,data);

}

return last;

}/\*End of create\_list()\*/

struct node \*addtoempty(struct node \*last,int data) {

struct node \*tmp;

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

tmp->info=data;

last=tmp;

last->link=last;

return last;

}/\*End of addtoempty( )\*/

struct node \*addatbeg(struct node \*last,int data) {

struct node \*tmp;

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

tmp->info=data;

tmp->link=last->link;

last->link=tmp;

return last;

}/\*End of addatbeg( )\*/

struct node \*addatend(struct node \*last,int data) {

struct node \*tmp;

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

tmp->info=data;

tmp->link=last->link;

last->link=tmp;

last=tmp;

return last;

}/\*End of addatend( )\*/

struct node \*addafter(struct node \*last,int data,int item) {

struct node \*tmp,\*p;

p=last->link;

do {

if(p->info==item) {

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

tmp->info=data;

tmp->link=p->link;

p->link=tmp;

if(p==last)

last=tmp;

return last;

}

p=p->link;

}while(p!=last->link);

printf("%d not present in the list\n",item);

return last;

}/\*End of addafter()\*/

struct node \*addbefore(struct node \*last,int data,int item) {

struct node \*tmp,\*p;

if(last==NULL ) {

printf("List is empty\n");

return last;

}

if(last->link->info==item) {

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

tmp->info=data;

tmp->link=last->link;

last->link=tmp;

return last;

}

p=last->link;

do {

if(p->link->info==item) {

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

tmp->info=data;

tmp->link=p->link;

p->link=tmp;

return last;

}

p=p->link;

}while(p!=last->link);

printf("%d not present in list\n",item);

return last;

}/\*End of addbefore()\*/

struct node \*del(struct node \*last,int data) {

struct node \*tmp,\*p;

if(last==NULL) {

printf("List is empty\n");

return last;

}

/\*Deletion of only node\*/

if(last->link==last && last->info==data) {

tmp=last;

last=NULL;

free(tmp);

return last;

}

/\*Deletion of first node\*/

if(last->link->info==data) {

tmp=last->link;

last->link=tmp->link;

free(tmp);

return last;

}

/\*Deletion in between\*/

p=last->link;

while(p->link!=last) {

if(p->link->info==data) {

tmp=p->link;

p->link=tmp->link;

free(tmp);

return last;

}

p=p->link;

}

/\*Deletion of last node\*/

if(last->info==data) {

tmp=last;

p->link=last->link;

last=p;

free(tmp);

return last;

}

printf("Element %d not found\n",data);

return last;

}/\*End of del( )\*/

void display(struct node \*last) {

struct node \*p;

if(last==NULL) {

printf("List is empty\n");

return;

}

printf("List is:\n");

p=last->link;

do {

printf("%d ",p->info);

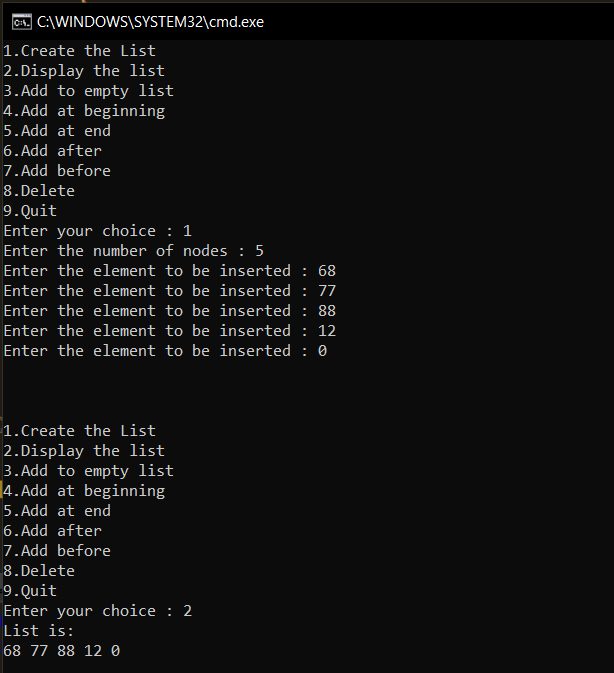
p=p->link;

}while(p!=last->link);

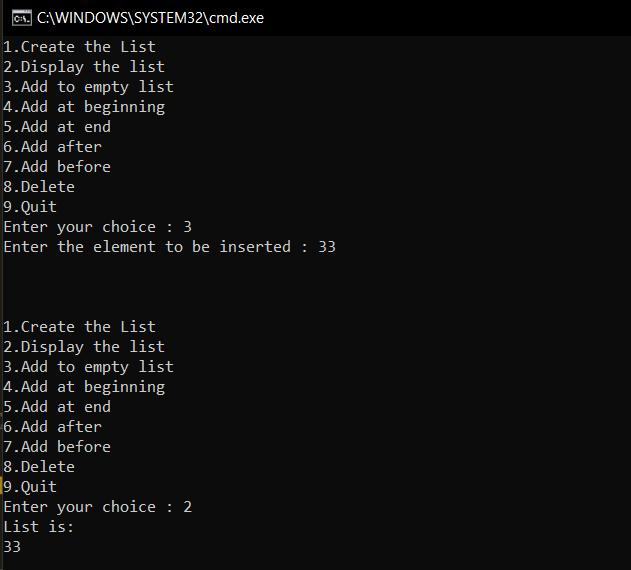
printf("\n");

}

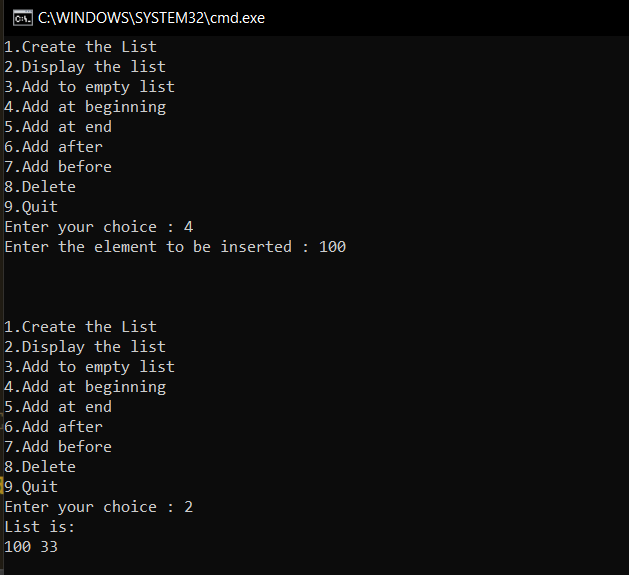
Output 1:



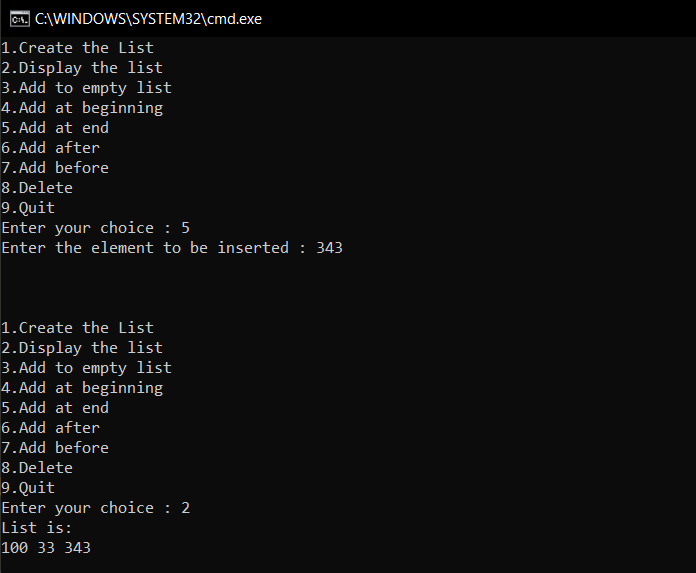
Output 2:



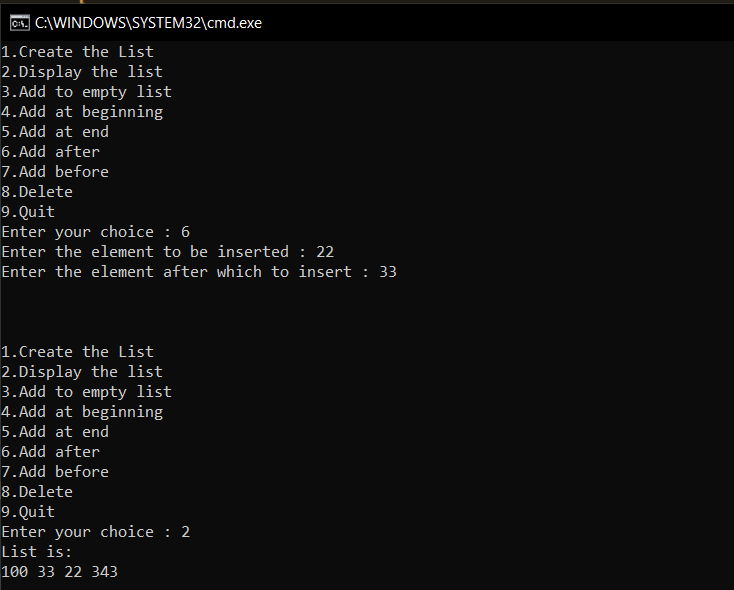
Output 3:



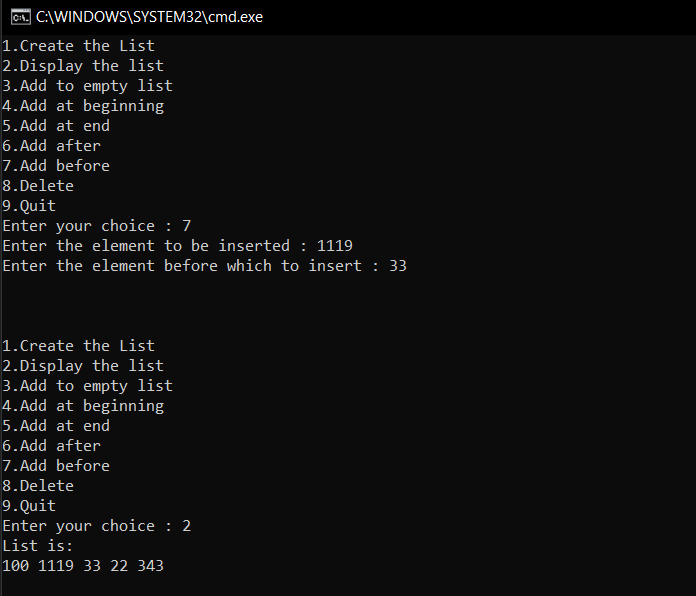
Output 4:



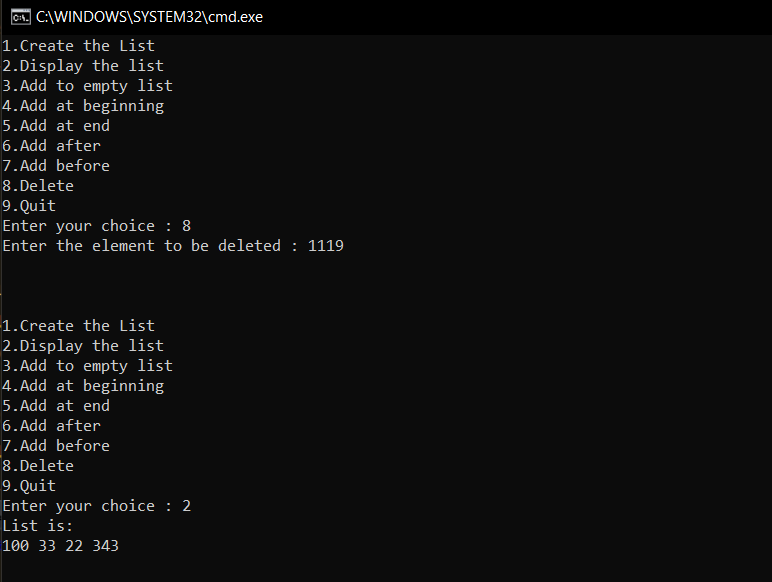
Output 5:



Output 6:



Output 7:



3. Write a program to add two polynomials using linked lists

Program:

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node {

float coef;

int expo;

struct node \*link;

};

struct node \*create(struct node \*);

struct node \*insert\_s(struct node \*,float,int);

struct node \*insert(struct node \*,float,int);

void display(struct node \*ptr);

void poly\_add(struct node \*,struct node \*);

int main( ) {

struct node \*start1=NULL,\*start2=NULL;

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Enter polynomial 1 :\n");

start1=create(start1);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Enter polynomial 2 :\n");

start2=create(start2);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");

printf("Polynomial 1 is : ");

display(start1);

printf("Polynomial 2 is : ");

display(start2);

poly\_add(start1, start2);

getch();

return 0;

}/\*End of main()\*/

struct node \*create(struct node \*start) {

int i,n,ex;

float co;

printf("Enter the number of terms : ");

scanf("%d",&n);

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

printf("Enter coeficient for term %d : ",i);

scanf("%f",&co);

printf("Enter exponent for term %d : ",i);

scanf("%d",&ex);

start=insert\_s(start,co,ex);

}

return start;

}/\*End of create()\*/

struct node \*insert\_s(struct node \*start,float co,int ex) {

struct node \*ptr,\*tmp;

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

tmp->coef=co;

tmp->expo=ex;

/\*list empty or exp greater than first one \*/

if(start==NULL || ex > start->expo) {

tmp->link=start;

start=tmp;

}

else {

ptr=start;

while(ptr->link!=NULL && ptr->link->expo >= ex) ptr=ptr->link;

tmp->link=ptr->link;

ptr->link=tmp;

}

return start;

}/\*End of insert()\*/

struct node \*insert(struct node \*start,float co,int ex) {

struct node \*ptr,\*tmp;

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

tmp->coef=co;

tmp->expo=ex;

/\*If list is empty\*/

if(start==NULL) {

tmp->link=start;

start=tmp;

}

else { /\*Insert at the end of the list\*/

ptr=start;

while(ptr->link!=NULL)

ptr=ptr->link;

tmp->link=ptr->link;

ptr->link=tmp;

}

return start;

}/\*End of insert()\*/

void display(struct node \*ptr) {

if(ptr==NULL) {

printf("Zero polynomial\n");

return;

}

while(ptr!=NULL) {

printf("(%.1fx^(%d))", ptr->coef,ptr->expo);

ptr=ptr->link;

if(ptr!=NULL) printf(" + ");

else printf("\n");

}

}/\*End of display()\*/

void poly\_add(struct node \*p1,struct node \*p2) {

struct node \*start3;

start3=NULL;

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

if(p1->expo > p2->expo) {

start3=insert(start3,p1->coef,p1->expo);

p1=p1->link;

}

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

start3=insert(start3,p2->coef,p2->expo);

p2=p2->link;

}

else if(p1->expo==p2->expo) {

start3=insert(start3,p1->coef+p2->coef,p1->expo);

p1=p1->link;

p2=p2->link;

}

}

/\*if poly2 has finished and elements left in poly1\*/

while(p1!=NULL) {

start3=insert(start3,p1->coef,p1->expo);

p1=p1->link;

}

/\*if poly1 has finished and elements left in poly2\*/

while(p2!=NULL) {

start3=insert(start3,p2->coef,p2->expo);

p2=p2->link;

}

printf("Added polynomial is : ");

display(start3);

}

Output :

