Name-Ayan Sadhukhan

Regd No-19BEC1448

*Question 1:*

* 1. Write a C program to find local maximum elements a doubly linked list. An element is a local maximum, if it's greater than both its neighbors. In above list 34 and 78 are local maximum.

*Code:*

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

struct Node \*prev;

}node;

void insert(node \*\*head\_addr,node \*\*tail\_addr, int data)

{

// Use tail pointer to insert at the end

// Allocate memory for the new node

node \*newnode;

newnode=(node \*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

newnode->prev=NULL;

//if the list is empty

if(\*head\_addr==NULL)

{

\*head\_addr=newnode;

\*tail\_addr=newnode;

}

else

{

(\*tail\_addr)->next=newnode;

newnode->prev=\*tail\_addr;

\*tail\_addr=newnode;

}

}

void \*del\_last(node \*\*head\_addr,node\*\* tail\_addr)

{

node \*temp=\*tail\_addr;

if(temp!=NULL)

{

if(\*head\_addr==\*tail\_addr)

\*head\_addr=\*tail\_addr=NULL;

else

{

(\*tail\_addr)->prev->next=NULL;

\*tail\_addr=(\*tail\_addr)->prev;

}

free(temp);

}

else

printf("\n list is empty\n");

}

void printforward(node \*head)

{

if(head==NULL)

return;

printf("%d\n", head->data);

printforward(head->next);

return;

}

void printbackward(node \*tail)

{

if(tail==NULL)

return;

printf("%d\n", tail->data);

printbackward(tail->prev);

}

void maximum(node \*head)

{

node \*p=NULL;

p=head->next;

while(p->next!=NULL)

{

if((p->next->data<p->data)&&(p->prev->data<p->data))

{

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

}

p=p->next;

}

}

node \*find(node\* head, int value)

{

while(head!=NULL && head->data!=value)

head=head->next;

return head;

}

int main()

{

int data,choice, printoption;

// head always points to the first node of the linked list

// tail always points to the last node of the linked list

node \*head,\*tail,\*temp;

tail=head=NULL;

do

{

printf("1. Insert \n");

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

printf("3.Find\n");

printf("4.Delete last\n");

printf("5.Find the max");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the data to be inserted\n");

scanf("%d",&data);

insert(&head,&tail,data);

break;

case 2:

printf("1.forward?\n 2. backward?\n");

scanf("%d",&printoption);

if(printoption==1)

printforward(head);

else

printbackward(tail);

break;

case 3:

printf("Enter the data to be found\n");

scanf("%d",&data);

temp=find(head,data);

if(temp==NULL)

printf("data is not found\n");

else

printf("data %d is found\n", temp->data);

break;

case 4:

del\_last(&head,&tail);

break;

case 5:

maximum(head);

}

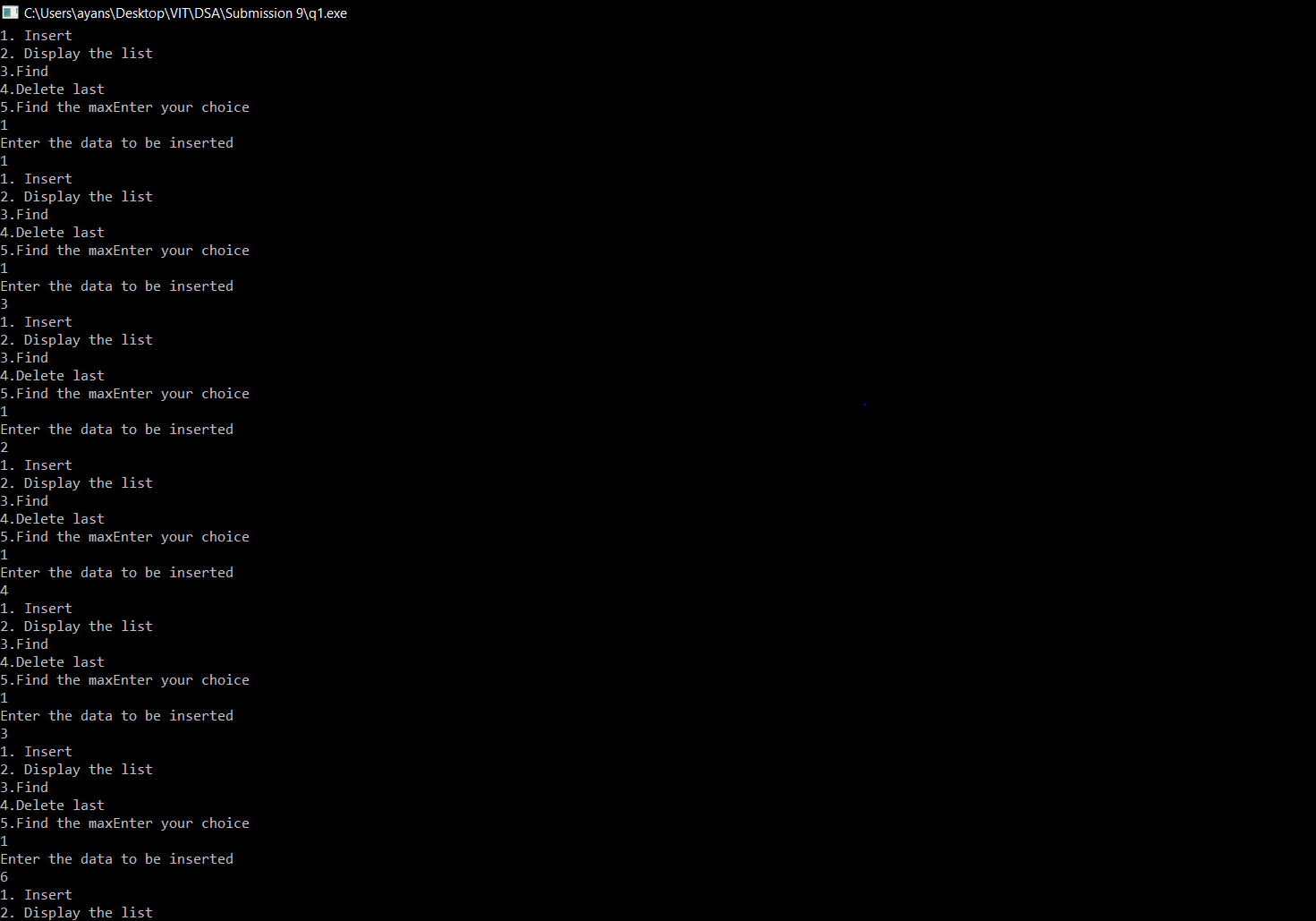
}

while(choice<=4);

return 0;

}

*Output:*





*Question 2:*

* 1. Write a C program, which inserts an element between every pair of consecutive elements. The new element is the sum of neighbor. In above case 12 46 34 90 56 134 78 93 15 80 65 137 72.

*Code:*

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

struct Node \*prev;

}node;

void insert(node \*\*head\_addr,node \*\*tail\_addr, int data)

{

node \*newnode;

newnode=(node \*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

newnode->prev=NULL;

if(\*head\_addr==NULL)

{

\*head\_addr=newnode;

\*tail\_addr=newnode;

}

else

{

(\*tail\_addr)->next=newnode;

newnode->prev=\*tail\_addr;

\*tail\_addr=newnode;

}

}

void \*del\_last(node \*\*head\_addr,node\*\* tail\_addr)

{

node \*temp=\*tail\_addr;

if(temp!=NULL)

{

if(\*head\_addr==\*tail\_addr)

\*head\_addr=\*tail\_addr=NULL;

else

{

(\*tail\_addr)->prev->next=NULL;

\*tail\_addr=(\*tail\_addr)->prev;

}

free(temp);

}

else

printf("\n list is empty\n");

}

node \*InsBet(node \*head) {

node \*i,\*j,\*t;

i=head;

while(i->next!=NULL) {

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

t->data = i->data + i->next->data;

t->next = i->next; i->next->prev = t;

i->next = t; t->prev = i;

i = t->next;

}

return head;

}

void printforward(node \*head)

{

if(head==NULL)

return;

printf("%d\n", head->data);

printforward(head->next);

return;

}

void printbackward(node \*tail)

{

if(tail==NULL)

return;

printf("%d\n", tail->data);

printbackward(tail->prev);

}

node \*find(node\* head, int value)

{

while(head!=NULL && head->data!=value)

head=head->next;

return head;

}

int main()

{

int data,choice, printoption;

// head always points to the first node of the linked list

// tail always points to the last node of the linked list

node \*head,\*tail,\*temp;

tail=head=NULL;

do

{

printf("1. Insert \n");

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

printf("3.Find\n");

printf("4.Delete last\n");

printf("5.Enter the sum of consecutive elements in the dll");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the data to be inserted\n");

scanf("%d",&data);

insert(&head,&tail,data);

break;

case 2:

printf("1.forward?\n 2. backward?\n");

scanf("%d",&printoption);

if(printoption==1)

printforward(head);

else

printbackward(tail);

break;

case 3:

printf("Enter the data to be found\n");

scanf("%d",&data);

temp=find(head,data);

if(temp==NULL)

printf("data is not found\n");

else

printf("data %d is found\n", temp->data);

break;

case 4:

del\_last(&head,&tail);

break;

case 5:

node \*p=NULL;

p=InsBet(head);

printforward(p);

}

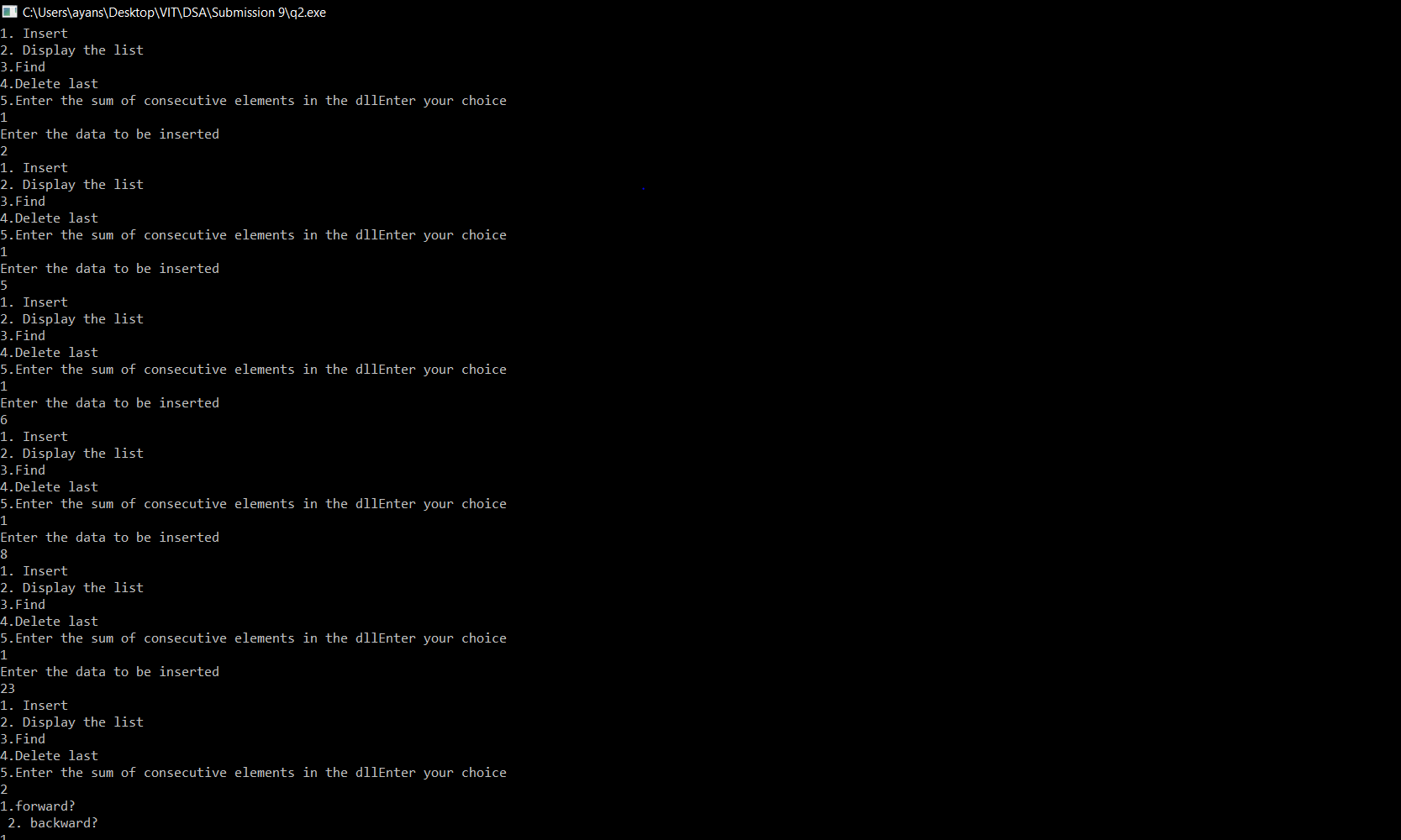
}

while(choice<=5);

return 0;

}

*Output:*





*Question 3:*

* 1. Read two locations. Write a C program to delete all elements between them. e.g. if linked list is 12 34 56 78 15 65 72 and locations are 2 and 5 then final linked list is 12 65 72.

*Code:*

#include <stdio.h>

#include <stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

struct Node \*prev;

}node;

void insert(node \*\*head\_addr,node \*\*tail\_addr, int data)

{

// Use tail pointer to insert at the end

// Allocate memory for the new node

node \*newnode;

newnode=(node \*)malloc(sizeof(node));

newnode->data = data;

newnode->next = NULL;

newnode->prev=NULL;

//if the list is empty

if(\*head\_addr==NULL)

{

\*head\_addr=newnode;

\*tail\_addr=newnode;

}

else

{

(\*tail\_addr)->next=newnode;

newnode->prev=\*tail\_addr;

\*tail\_addr=newnode;

}

}

void \*del\_last(node \*\*head\_addr,node\*\* tail\_addr)

{

node \*temp=\*tail\_addr;

if(temp!=NULL)

{

if(\*head\_addr==\*tail\_addr)

\*head\_addr=\*tail\_addr=NULL;

else

{

(\*tail\_addr)->prev->next=NULL;

\*tail\_addr=(\*tail\_addr)->prev;

}

free(temp);

}

else

printf("\n list is empty\n");

}

void printforward(node \*head)

{

if(head==NULL)

return;

printf("%d\n", head->data);

printforward(head->next);

return;

}

void printbackward(node \*tail)

{

if(tail==NULL)

return;

printf("%d\n", tail->data);

printbackward(tail->prev);

}

node \*find(node\* head, int value)

{

while(head!=NULL && head->data!=value)

head=head->next;

return head;

}

node \*del(node \*head,int a,int b)

{

node \*p=NULL;

node \*p1=NULL;

p=head;

p1=head;

int f=0,k=0;

while(p->next!=NULL)

{

f=f+1;

if(f==a)

{

p=p->prev;

break;

}

p=p->next;

}

while(p1->next!=NULL)

{

k=k+1;

if(k==b)

{

p1=p1->next;

break;

}

p1=p1->next;

}

p->next=p1;

p1->prev=p;

return head;

}

int main()

{

int data,choice, printoption;

// head always points to the first node of the linked list

// tail always points to the last node of the linked list

node \*head,\*tail,\*temp;

tail=head=NULL;

do

{

printf("1. Insert \n");

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

printf("3.Find\n");

printf("4.Delete last\n");

printf("5.To Delete the middle elements \n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("Enter the data to be inserted\n");

scanf("%d",&data);

insert(&head,&tail,data);

break;

case 2:

printf("1.forward?\n 2. backward?\n");

scanf("%d",&printoption);

if(printoption==1)

printforward(head);

else

printbackward(tail);

break;

case 3:

printf("Enter the data to be found\n");

scanf("%d",&data);

temp=find(head,data);

if(temp==NULL)

printf("data is not found\n");

else

printf("data %d is found\n", temp->data);

break;

case 4:

del\_last(&head,&tail);

break;

case 5:

int a;

int b;

printf("Enter the position from where you want to start deleting \n");

scanf("%d",&a);

printf("Enter the position till where you want to delete \n");

scanf("%d",&b);

node \*p=NULL;

p=del(head,a,b);

}

}

while(choice<=5);

return 0;

}

*Output:*



