*Name: Ayan Sadhukhan*

*Regd No:19BEC1448*

*Question 1:*

Write a C program to split the given linked list ( length >1) into two sublists — one for the front half, and one in the back half. If the number of elements is odd, the extra element should go on the front list. For example, a list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}.

*Code:*

#include <stdio.h>

#include <stdlib.h>

void insert();

void delet();

void display();

struct node

{

int data;

struct node \*link;

};

struct node \*first=NULL,\*last=NULL,\*next,\*prev,\*cur;

int nodecount=0;

int Length(struct node\* head)

{

int count = 0;

struct node\* current = head;

while (current != NULL)

{

count++;

current=current->link;

}

return count;

}

void FrontBackSplit(struct node\* source, struct node\*\* frontRef,

struct node\*\* backRef)

{

int len = Length(source);

if (len < 2)

{

\*frontRef = source;

\*backRef = NULL;

return;

}

struct node\* current = source;

int hopCount = (len - 1) / 2;

for (int i = 0; i < hopCount; i++)

current = current->link;

\*frontRef = source;

\*backRef = current->link;

current->link = NULL;

}

void printList(struct node\* head)

{

struct node\* ptr = head;

while (ptr)

{

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

ptr = ptr->link;

}

printf("null");

}

void insert()

{

int pos,c=1;

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

// printf("\nENTER THE DATA: ");

// scanf("%d",&cur->data);

// cur->link=NULL;

if(first==NULL)

{ printf("\nENTER THE DATA: ");

scanf("%d",&cur->data);

cur->link=NULL;

first=cur;

nodecount++;

}

else

{

printf("number of nodes in the list =%d \n",nodecount);

printf("\nENTER THE POSITION: 1->for front & any other value for any other position");

scanf("%d",&pos);

if(pos>nodecount+1)

{

printf("\n Enter valid position \n");

}

else

{ printf("\nENTER THE DATA: ");

scanf("%d",&cur->data);

if((pos==1) &&(first!=NULL))

{

cur->link = first;

first=cur;

}

else

{

next=first;

while(c<pos)

{

prev=next;

next=prev->link;

c++;

}

if(prev==NULL)

{

printf("\nINVALID POSITION\n");

}

else

{

cur->link=prev->link;

prev->link=cur;

}

}

nodecount++;

}

}

}

void delet()

{

int pos,c=1;

printf("\nENTER THE POSITION : ");

scanf("%d",&pos);

if(pos>nodecount)

{

printf("\n Enter valid position \n");

}

else

{

if(first==NULL)

{

printf("\nLIST IS EMPTY\n");

}

else if(pos==1 && first->link==NULL)

{

printf("\n DELETED ELEMENT IS %d\n",first->data);

free(first);

first=NULL;

}

else if(pos==1 && first->link!=NULL)

{

cur=first;

first=first->link;

cur->link=NULL;

printf("\n DELETED ELEMENT IS %d\n",cur->data);

free(cur);

}

else

{

next=first;

while(c<pos)

{

cur=next;

next=next->link;

c++;

}

cur->link=next->link;

next->link=NULL;

if(next==NULL)

{

printf("\nINVALID POSITION\n");

}

else

{

printf("\n DELETED ELEMENT IS %d\n",next->data);

free(next);

}

}

}

}

void display()

{

cur=first;

while(cur!=NULL)

{

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

cur=cur->link;

}

}

int main()

{

int ch;

printf("\n\nSINGLY LINKED LIST");

do

{

printf("\n\n 1.INSERT\n 2.DELETE\n 3.Split the linked list into two and print it.\n 4.EXIT");

printf("\n\nENTER YOUR CHOICE : ");

scanf("%d",&ch);

switch(ch)

{

case 1:

insert();

display();

break;

case 2:

delet();

display();

break;

case 3:

{struct node\* head = NULL;

struct node \*a = NULL, \*b = NULL;

FrontBackSplit(first, &a, &b);

printf("Front List : ");

printList(a);

printf("\nBack List : ");

printList(b);

break;}

case 4:

exit(0);

break;

default:

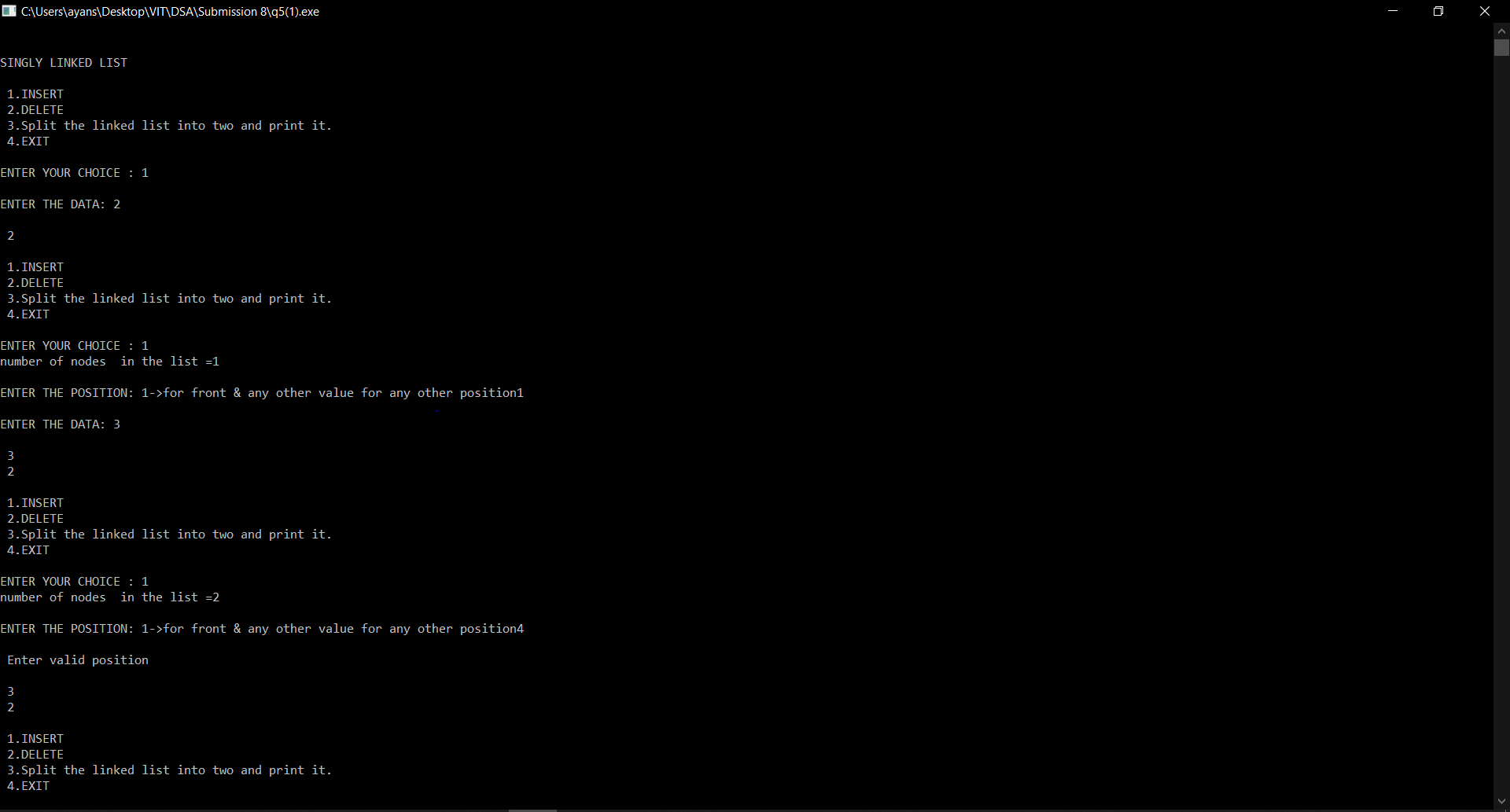
printf("Invalid choice...");

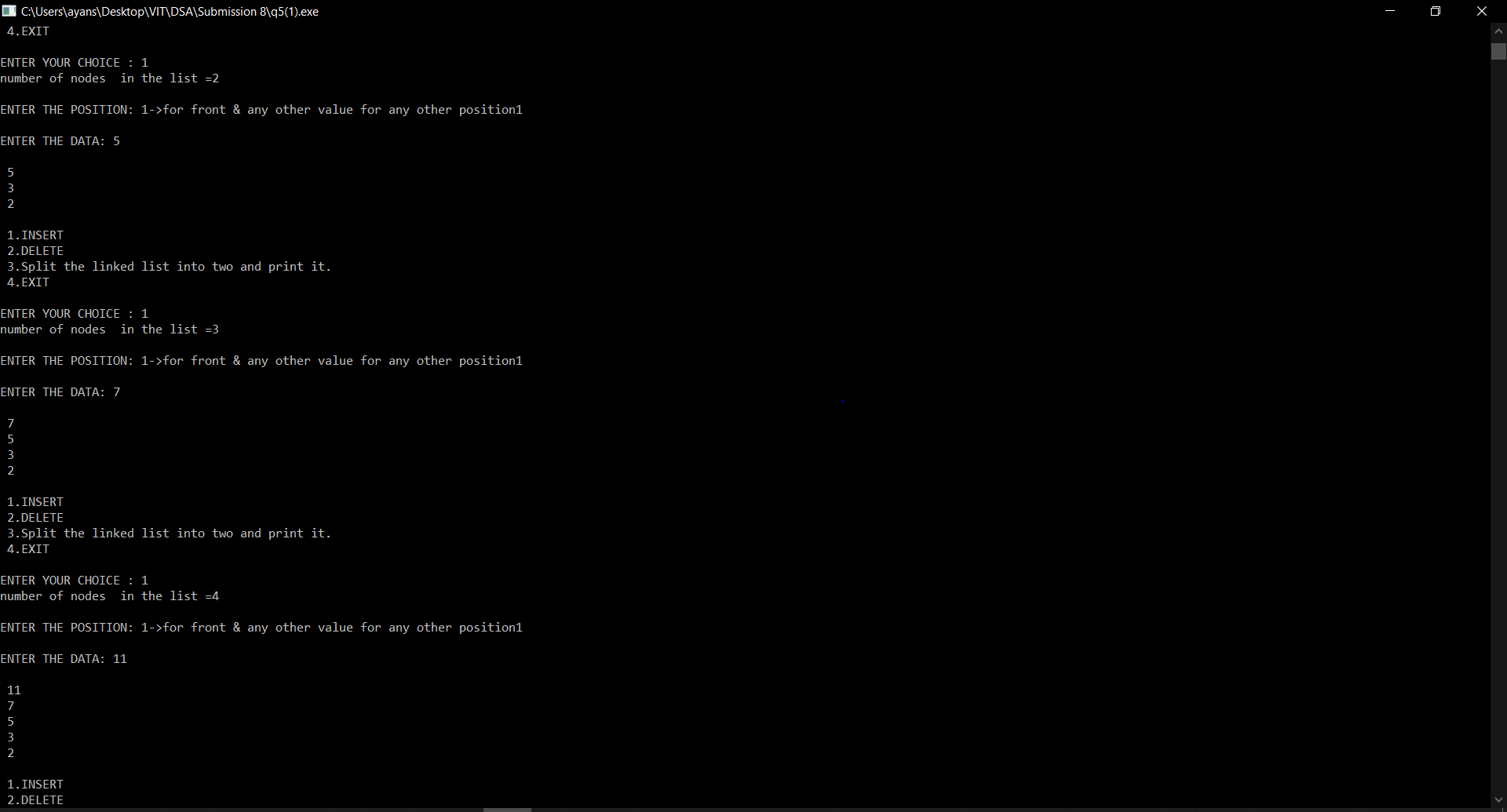
}

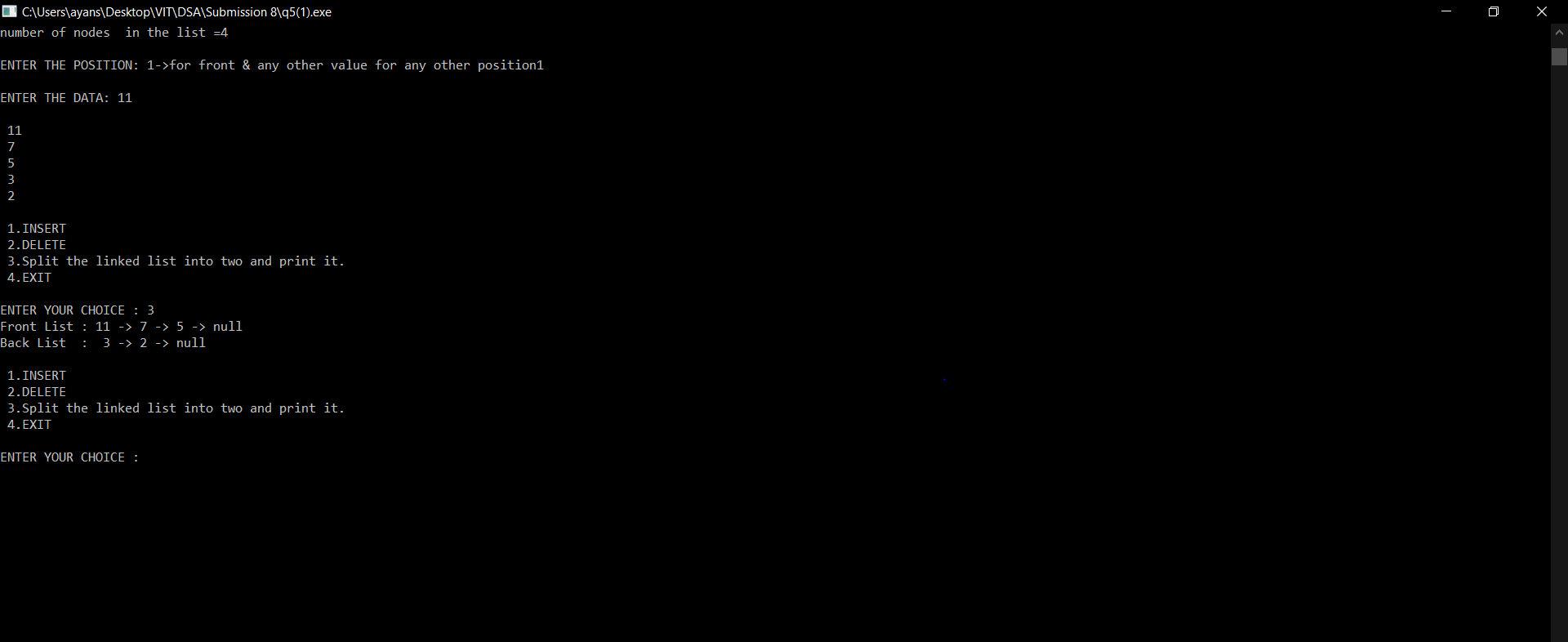
}while(1);

}

*Output:*







*Question 2:*

Given two sorted lists L1 and L2 , write a C program to compute L1∩L2 ( common elements in the two list in sorted order ) using only the basic list operations. For example, if L1= { 1, 3, 5, 7, 9, 12} and L2={ 3,7,17 } then L1∩L2 = { 3,7} .

*Code:*

#include <stdio.h>

#include <stdlib.h>

struct node{

struct node\*next;

int data;

};

struct node \* intersection(struct node \* L1, struct node\* L2){

if(L1 == NULL || L2 == NULL)

return NULL;

struct node \* output = NULL;

struct node \* outTail = NULL;

while(L1&&L2){

if(L1->data<L2->data){

L1 = L1->next;

}

else if(L2->data<L1->data){

L2 = L2->next;

}

else{

int data = L1->data;

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

newNode->data = data;

newNode->next = NULL;

if(output == NULL){

outTail = output = newNode;

}

else{

outTail->next = newNode;

outTail = outTail->next;

}

while(L1 && L2 && L1->data == data && L2->data == data){

L1 = L1->next;

L2 = L2->next;

}

}

}

return output;

}

struct node \* createList(int listNum){

struct node \* list = NULL;

struct node \* list\_tail = NULL;

printf("Enter elements of List %d in increasing order\n",listNum);

char ch = 'y';

do{

int data;

printf("Enter element : ");

scanf("%d",&data);

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

newNode->data = data;

newNode->next = NULL;

if(list == NULL){

list = list\_tail = newNode;

}

else{

list\_tail->next = newNode;

list\_tail = list\_tail->next;

}

printf("Would you like to insert another element [Y/N] : ");

scanf(" %c",&ch);

}while(ch == 'y' || ch == 'Y');

return list;

}

void print(struct node \* list){

if(list == NULL){

printf("Empty List\n");

return;

}

while(list!=NULL){

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

list = list->next;

}

printf("\n");

}

int main() {

struct node \* L1 = NULL;

struct node \* L2 = NULL;

struct node \* L4 = NULL;

L1 = createList(1);

L2 = createList(2);

printf("List 1 : ");

print(L1);

printf("List 2 : ");

print(L2);

printf("Intersection : ");

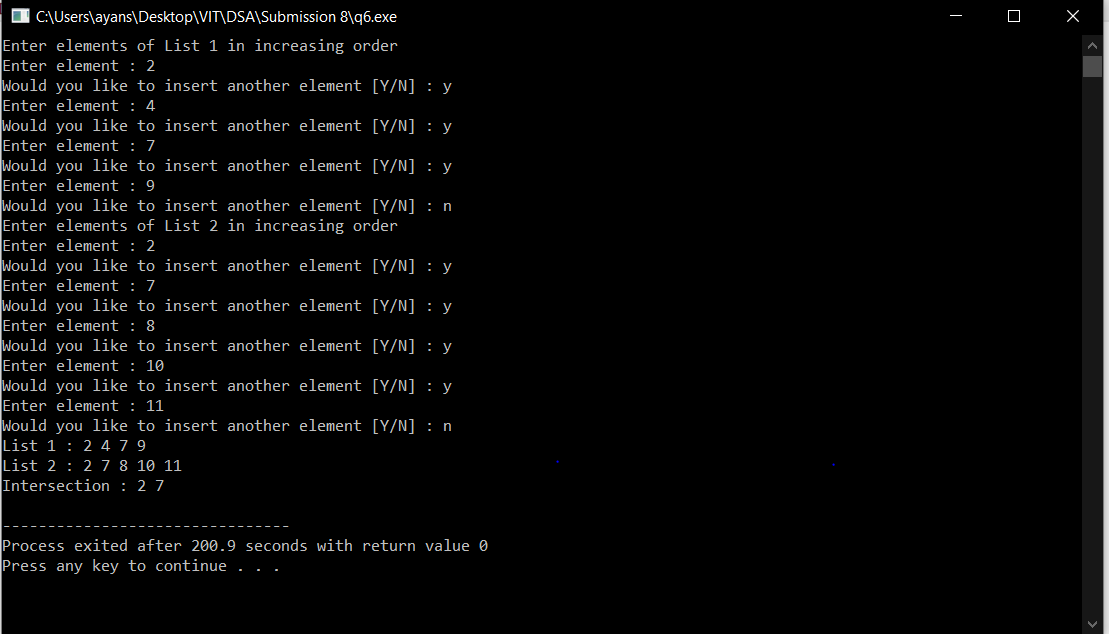
L4 = intersection(L1, L2);

print(L4);

return 0;

}

*Output:*



*Question 3:*

Given two lists L1 and L2 , write a C program to compute L1 - L2 ( remove the elements from the list L1, which are already present in L2 ) using only the basic list operations. For example, if L1= { 1, 9, 5, 7,1 9, 12} and L2={ 3,7,12 } then L1-L2 = {1, 9, 5,19} .

*Code:*

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

//using namespace std;

struct node

{

int data;

struct node \*link;

};

struct node \*createlist()

{

struct node\* list=NULL;

struct node \*end=NULL;

printf("Enter the elements in the list :\n");

char ch='Y';

do

{

int n;

printf("Enter the element :");

scanf("%d",&n);

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

newNode->data = n;

newNode->link = NULL;

if(list == NULL){

list = end = newNode;

}

else{

end->link = newNode;

end = end->link;

}

printf("Would you like to insert another element [Y/N] : ");

scanf(" %c",&ch);

}while(ch == 'y' || ch == 'Y');

return list;

}

void print(struct node \* list){

if(list == NULL){

printf("Empty List\n");

return;

}

while(list!=NULL){

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

list = list->link;

}

printf("\n");

}

struct node \*deletenode(struct node \*list,int d)

{

struct node \*p=NULL;

struct node \*temp=NULL;

if(list==NULL)

return list;

if(list->data==d)

{

temp=list;

list=list->link;

free(temp);

return list;

}

p=list;

while(p->link!=NULL)

{

if(p->link->data==d)

break;

p=p->link;

}

if(p->link!=NULL)

{

temp=p->link;

p->link=temp->link;

free(temp);

}

return list;

}

struct node\* remove(struct node \*list1,struct node \*list2)

{

struct node \*list3=NULL,\*p=NULL;

list3=list1;

p=list2;

while(p!=NULL)

{

list1=deletenode(list1,p->data);

p=p->link;

}

return list1;

}

int main()

{

struct node \*l1=NULL;

struct node \*l2=NULL;

printf("The lists before the subtraction is :\n");

l1=createlist();

l2=createlist();

print(l1);

print(l2);

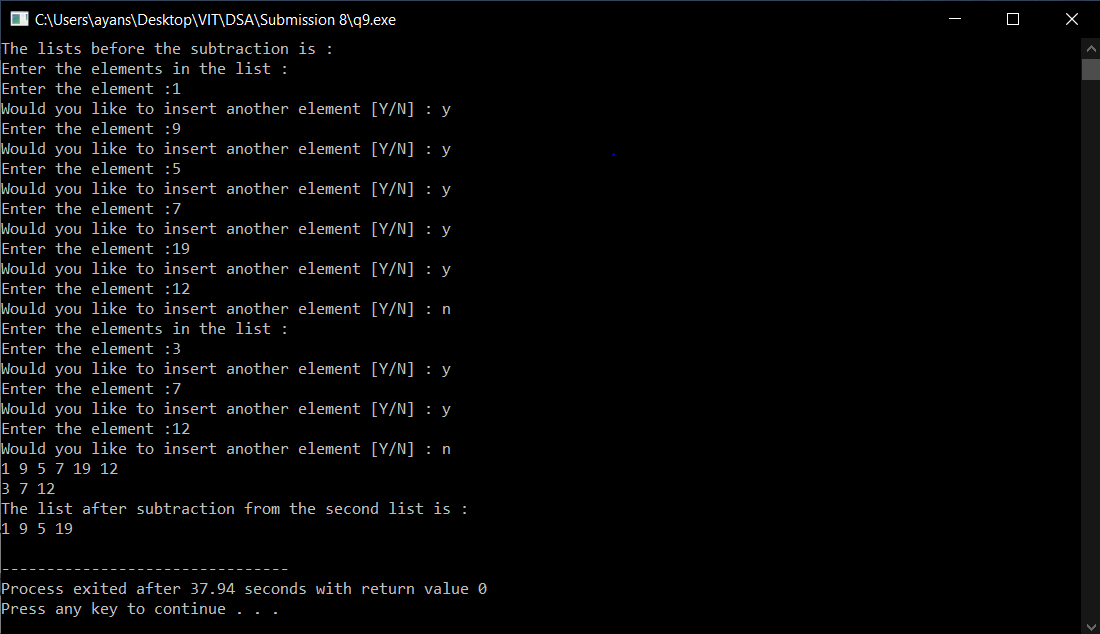
l1=remove(l1,l2);

printf("The list after subtraction from the second list is :\n");

print(l1);

}

*Output:*



*Question 4:*

Write a C program for ‘RemoveDuplicates()’ function which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.

*Code:*

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node\* next;

};

void removeDuplicates(struct Node\* head)

{

struct Node\* current = head;

struct Node\* next\_next;

if (current == NULL)

return;

while (current->next != NULL)

{

if (current->data == current->next->data)

{

next\_next = current->next->next;

free(current->next);

current->next = next\_next;

}

else

{

current = current->next;

}

}

}

void push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node =

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

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*node)

{

while (node!=NULL)

{

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

node = node->next;

}

}

void swap(struct Node \*a, struct Node \*b)

{

int temp = a->data;

a->data = b->data;

b->data = temp;

}

struct Node \*bubbleSort(struct Node \*start)

{

int swapped, i;

struct Node \*ptr1=NULL;

struct Node \*lptr = NULL;

/\* Checking for empty list \*/

if (start == NULL)

return start;

do

{

swapped = 0;

ptr1 = start;

while (ptr1->next != lptr)

{

if (ptr1->data > ptr1->next->data)

{

swap(ptr1, ptr1->next);

swapped = 1;

}

ptr1 = ptr1->next;

}

lptr = ptr1;

}

while (swapped);

return start;

}

/\* function to swap data of two nodes a and b\*/

int main()

{

struct Node\* head = NULL;

int n,m;

printf("Enter the number of elements to be entered :");

scanf("%d",&n);

for(int i=0;i<n;i++)

{

printf("Enter the element :");

scanf("%d",&m);

push(&head, m);

}

printf("\n Linked list before duplicate removal ");

printList(head);

printf("\n Linked list before sorting ");

head=bubbleSort(head);

printList(head);

printf("\n Linked list after sorting ");

printList(head);

removeDuplicates(head);

printf("\n Linked list after duplicate removal ");

printList(head);

return 0;

}

*Output:*

