**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**DATA STRUCTURES**

***Submitted by***

**RUSHIL BINDROO(1BM21CS172)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Oct 2022-Feb 2023**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “**DATA STRUCTURES**” carried out by **RUSHIL BINDROO (1BM21CS172),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab **- (22CS3PCDST)** work prescribed for the said degree.

VIKRANT.B.S               **Dr. Jyothi S Nayak**

Designation Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Experiment Title** | **Page No.** |
| 1 | STACK IMPLEMENTATION USING ARRAYS | 5 |
| 2 | CONVERSION OF INFIX TO POSTFIX EXPRESSION | 9 |
| 3 | TO DEMONSTRATE QUEUES USING ARRAYS | 15 |
| 4 | TO DEMONSTRATE CIRCULAR QUEUES USING ARRAYS | 20 |
| 5 | Singly Linked list -Insert | 27 |
| 6 | Singly Linked list -Insert | 35 |
| 7 | Sorting, Reversing and Concatenation of singly linked list | 42 |
| 8 | Implement Stack & Queues using Linked Representation | 54 |
| 9 | Implement doubly link list with primitive operations | 65 |
| 10 | BINARY SEARCH TREE | 77 |

**Course Outcome**

|  |  |
| --- | --- |
| CO1 | Analyze data structure operations for a given problem. |
| CO2 | Apply the concept of linear and nonlinear data structures. |
| CO3 | Conduct practical experiments for demonstrating the operations of different data structures. |
| CO4 | Design and develop solutions using the operations of linear and nonlinear data structure for a given specification. |

**LAB PROGRAM 1: STACK IMPLEMENTATION USING ARRAYS**

**Q : Write a program to simulate the working of stack using an array with the following: a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow**

**Code :**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

int top = -1, item, s[10];

void push(){

int b;

if(top>=10)

{

printf("Stack Overflow");

return;

}

printf("Enter no:");

scanf("%d",&b);

top++;

s[top]=b;

}

void pop(){

if(top==-1)

{

printf("Stack Underflow");

return;

}

printf("Deleted no=%d\n",s[top--]);

}

void display(){

if(top==-1)

{

printf("Stack is Empty");

return;

}

for(int i=top;i>-1;i--){

printf("%d\t",s[i]);

}

}

int main()

{

int choice;

while(1){

printf("\nEnter choice\n1.Push\n2.Pop\n3.Display\n4.Exit");

scanf("%d",&choice);

switch(choice)

{

case 1:push();

break;

case 2:pop();

break;

case 3:display();

break;

case 4:exit(0);

default :printf("Invalid chocie");

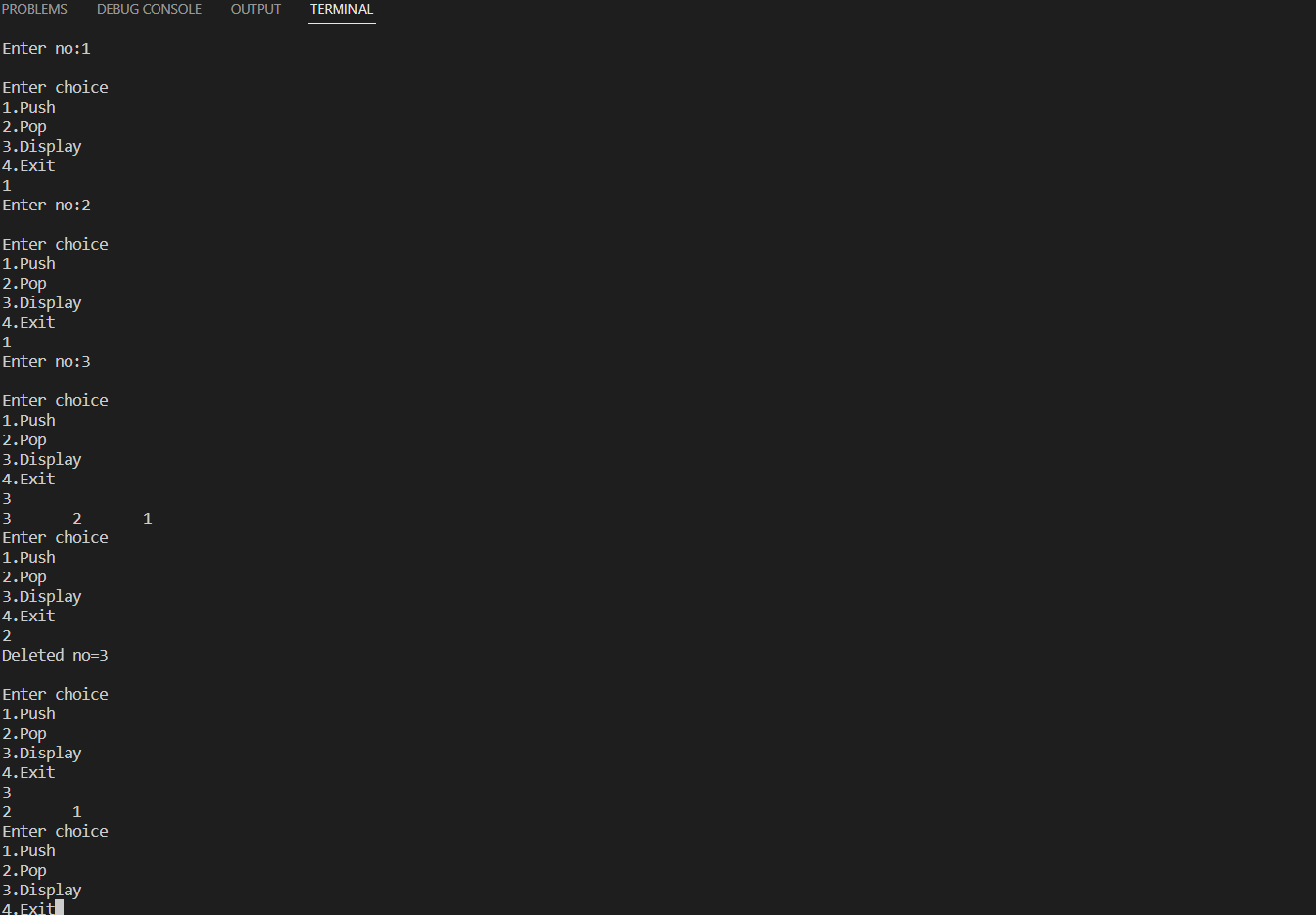
}

}

return 0;

}

**Output :**

****

**LAB PROGRAM 2 : CONVERSION OF INFIX TO POSTFIX EXPRESSION**

**Q : WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide)**

**Code :**

#include<stdio.h>

#include<conio.h>

#include<string.h>

int index=0,pos=0,top=-1,length;

char symbol, temp, infix[20], postfix[20], stack[20];

void infix\_postfix();

void push(char);

char pop();

int pred(char symb);

void main()

{

printf("Enter infix Expression");

scanf("%s",infix);

infix\_postfix();

printf("Infix Expression is %s",infix);

printf("Postfix Expression is %s",postfix);

getch();

}

void infix\_postfix()

{

length=strlen(infix);

push('#');

while(index<length)

{

symbol=infix[index];

switch(symbol)

{

case '(':push(symbol);

break;

case ')':temp=pop();

while(temp!='(')

{

postfix[pos]=temp;

pos++;

temp=pop();

}

break;

case'+':

case'-':

case'\*':

case'/':

case'^': while(pred(stack[top])>=pred(symbol))

{

temp=pop();

postfix[pos++]=temp;

}

push(symbol);

break;

default:postfix[pos++]=symbol;

}

index++;

}

while(top>0)

{

temp=pop();

postfix[pos++]=temp;

}

}

void push(char symbol)

{

top=top+1;

stack[top]=symbol;

}

char pop()

{

char symb;

symb=stack[top];

top=top-1;

return(symb);

}

int pred(char symbol)

{

int p;

switch(symbol)

{

case'^':p=3;

break;

case '\*':

case'/':p=2;break;

case '+':

case'-':p=1;break;

case'(':

p=0;break;

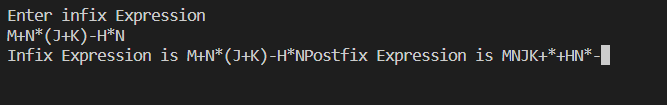
case'#':p=-1;break;

}

return(p);

}

**OUTPUT :**

****

**LAB PROGRAM 3 : TO DEMONSTRATE QUEUES USING ARRAYS**

**Q : WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions.**

**Code :**

#include <stdio.h>

# define size 3

void insert();

void delete1();

void dispaly();

int array[size];

int Rear = - 1, Front = - 1;

main()

{

int choice;

while (1)

{

printf("\n1.insert\n2.delete\n3.Display\n4.Exit\n");

printf("\nEnter your choice:\n");

scanf("%d", &choice);

switch (choice)

{

case 1: insert();

break;

case 2: delete1();

break;

case 3: display();

break;

case 4: exit(0);

default:

printf("\nIncorrect choice \n");

}

}

}

void insert()

{

int item;

if (Rear == size- 1)

printf("\nqueue overflow \n");

else

{

if (Front == - 1)

Front = 0;

printf("\nenter element to be inserted:\n ");

scanf("%d", &item);

Rear = Rear + 1;

array[Rear] = item;

}

}

void delete1()

{

if (Front == - 1 || Front > Rear)

{

printf("\nqueue underflow\n");

return ;

}

else

{

printf("\nElement deleted is: %d\n", array[Front]);

Front = Front + 1;

}

}

void display()

{

int i;

if (Front == - 1)

printf("\nEmpty Queue\n");

else

{

printf("\nQueue:\n");

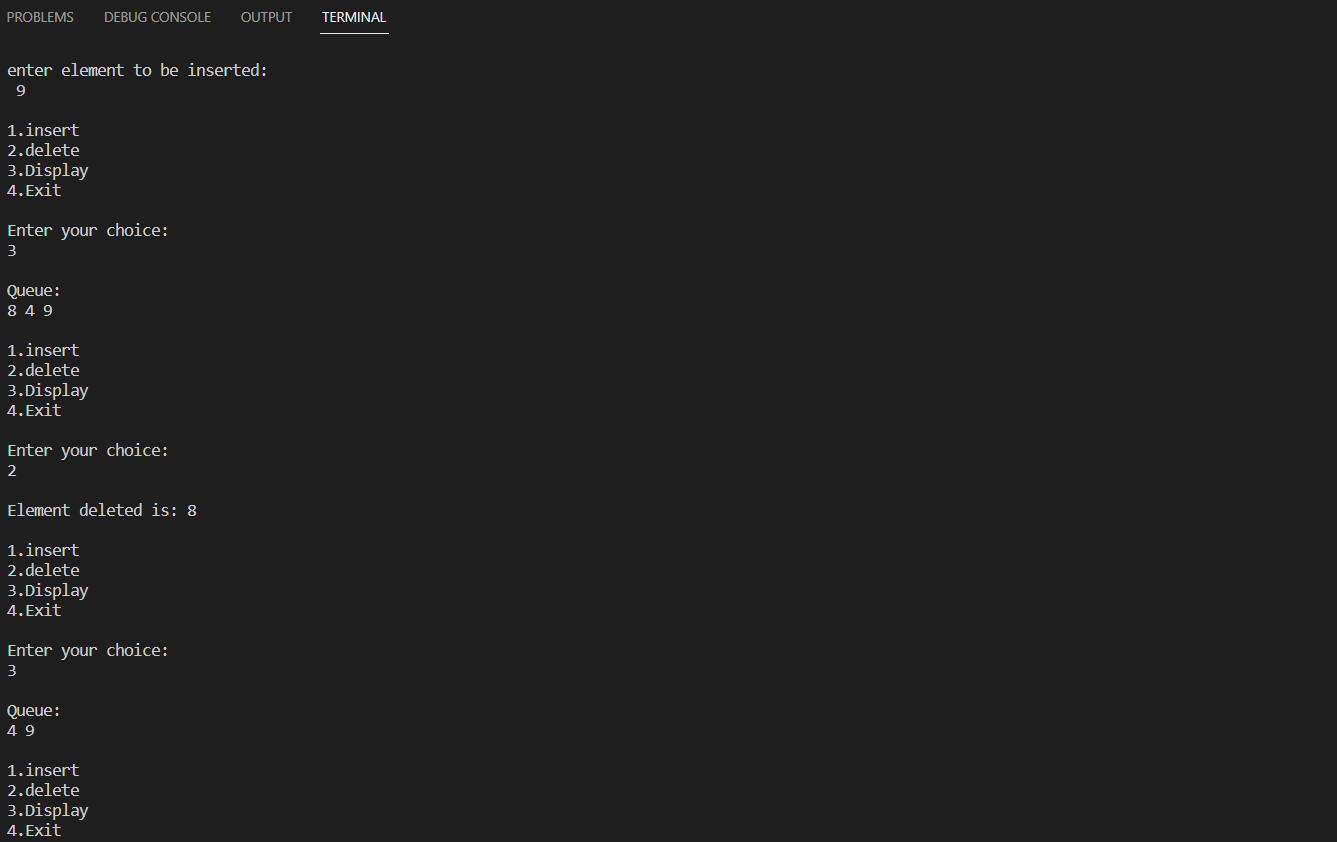
for (i= Front; i <= Rear; i++)

printf("%d ",array[i]);

printf("\n");

}

}

**Output : **

**LAB PROGRAM 4 :**

**Q : WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions**

**Code :**

#include<stdio.h>

# define MAX 3

void insert(int cqueue\_arr[],int item,int \*rear,int \*front)

{

if((\*front == 0 && \*rear == MAX-1) || (\*front == (\*rear)+1))

{

printf("\nQueue Overflow n");

return;

}

if(\*front == -1)

{

\*front = 0;

\*rear = 0;

}

else

{

if(\*rear == MAX-1)

\*rear = 0;

else

\*rear = (\*rear)+1;

}

cqueue\_arr[\*rear] = item ;

}

void deletion(int cqueue\_arr[],int \*rear,int \*front)

{

if(\*front == -1)

{

printf("\nQueue Underflown");

return ;

}

printf("\nElement deleted from queue is : %d",cqueue\_arr[\*front]);

if(\*front == \*rear)

{

\*front = -1;

\*rear=-1;

}

else

{

if(\*front == MAX-1)

\*front = 0;

else

\*front = (\*front)+1;

}

}

void display(int cqueue\_arr[],int \*rear,int \*front)

{

int front\_pos = \*front,rear\_pos = \*rear;

if(\*front == -1)

{

printf("\nQueue is empty");

return;

}

printf("\nQueue elements :");

if( front\_pos <= rear\_pos )

while(front\_pos <= rear\_pos)

{

printf("\n %d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

else

{

while(front\_pos <= MAX-1)

{

printf("\n%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

front\_pos = 0;

while(front\_pos <= rear\_pos)

{

printf("\n %d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

}

}

int main()

{

int choice,item;

int cqueue\_arr[MAX];

int front = -1;

int rear = -1;

do

{

printf("\n1.Insert\n");

printf("2.Delete\n");

printf("3.Display\n");

printf("4.Quit\n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1 :

printf("\nInput the element for insertion in queue : ");

scanf("%d", &item);

insert(cqueue\_arr,item,&rear,&front);

break;

case 2 :

deletion(cqueue\_arr,&rear,&front);

break;

case 3:

display(cqueue\_arr,&rear,&front);

break;

case 4:

break;

default:

printf("\nWrong choice");

}

}while(choice!=4);

return 0;

}

**OUTPUT :** ****

**LAB PROGRAM 5 : Singly linked list - Insert**

**Q : WAP to Implement Singly Linked List with following operations**

**a) Create a linked list.**

**b) Insertion of a node at first position, at any position and at end of list.**

**c) Display the contents of the linked list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

void create();

void display();

void insert\_head();

void insert\_last();

void insert\_val();

struct Node

{

int data;

struct Node \*link;

};

typedef struct Node node;

node \*start=NULL;

int main()

{

int ch;

while(1)

{

printf("\n1.Create\n2.Display \n3.Insert Head \n4.Insert Last\n5.Insert val\n6.Exit");

printf("\nEnter your choice:\n");

scanf("%d",&ch);

switch(ch)

{

case 1:create();

break;

case 2:display();

break;

case 3:insert\_head();

break;

case 4:insert\_last();

break;

case 5: insert\_val();

break;

case 6:exit(1);

default : printf("Invalid choice\n");

}

}

return 0;

}

void create()

{

int c;

node \*neww,\*curr;

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

curr=start;

printf("Enter element\n");

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

while(1)

{

printf("Do you want to add another element(1/0)\n");

scanf("%d",&c);

if(c==1)

{

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

printf("Enter element\n");

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

curr->link = neww;

curr=neww;

}

else

{

curr->link=NULL;

break;

}

}

}

void display()

{

node \*temp;

if(start==NULL)

{

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

return;

}

temp=start;

while(temp!=NULL)

{

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

temp = temp->link;

}

}

void insert\_head(){

node \*temp,\*mew;

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

temp = start;

printf("enter element value");

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

mew->link = start;

start = mew;

}

void insert\_last(){

node \*neww,\*temp;

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

temp = start;

printf("enter element value");

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

while(temp->link!=NULL)

{

temp = temp->link;

}

temp->link = neww;

neww->link = NULL;

}

void insert\_val(){

int pos;

node \*neww, \*temp;

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

printf("\nEnter element: ");

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

printf("Enter position\n");

scanf("%d",&pos);

if(pos==1)

{

neww->link=start;

start=neww;

return;

}

int i=1;

temp=start;

while(i<(pos-1) && temp!=NULL)

{

temp=temp->link;

i++;

}

if(i==(pos-1))

{

neww->link=temp->link;

temp->link=neww;

return;

}

if(temp==NULL)

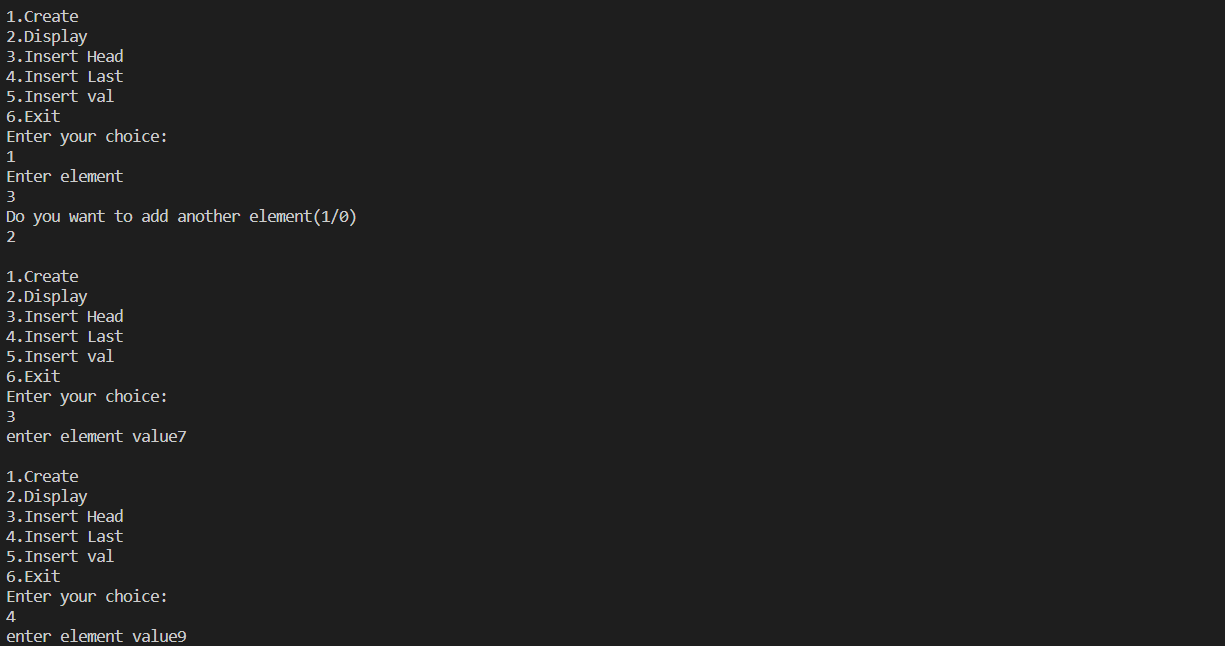
{

printf("Invalid position\n");

}

}

**Output :**

****

**LAB PROGRAM 6 : Singly linked list - Delete**

**Q : WAP to Implement Singly Linked List with following operations**

**a) Create a linked list.**

**b) Deletion of first element, specified element and last element in the list.**

**c) Display the contents of the linked list.**

**Code :**

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

void create();

void display();

void delete\_head();

void delete\_last();

void delete\_val();

struct Node

{

int data;

struct Node \*link;

};

typedef struct Node node;

node \*start=NULL;

int main()

{

int ch;

while(1)

{

printf("\n1.Create\n2.Display\n3.Delete Head\n4.Delete Last\n5.Delete val\n6.Exit");

printf("\nEnter your choice:\n");

scanf("%d",&ch);

switch(ch)

{

case 1: create();

break;

case 2:display();

break;

case 3: delete\_head();

break;

case 4:delete\_last();

break;

case 5:delete\_val();

break;

case 6:exit(1);

default: printf("Invalid choice\n");

}

}

return 0;

}

void create()

{

int c;

node \*neww,\*curr;

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

curr=start;

printf("Enter element\n");

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

while(1)

{

printf("Do you want to add another element(1/0)\n");

scanf("%d",&c);

if(c==1)

{

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

printf("Enter element\n");

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

curr->link = neww;

curr=neww;

}

else

{

curr->link=NULL;

break;

}

}

}

void display()

{

node \*temp;

if(start==NULL)

{

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

return;

}

temp=start;

while(temp!=NULL)

{

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

temp = temp->link;

}

}

void delete\_head(){

node \*ptr;

ptr = start;

start=start->link;

free(ptr);

}

void delete\_last(){

node \*ptr,\*prevptr;

ptr = start;

prevptr = start;

while(ptr->link != NULL)

{

prevptr = ptr;

ptr = ptr->link;

}

prevptr->link = NULL;

free(ptr);

}

void delete\_val(){

int val;

node \*ptr,\*prevptr;

prevptr = start;

ptr = start;

printf("enter value to be deleted");

scanf("%d",&val);

while(ptr->data!=val){

prevptr = ptr;

ptr = ptr->link;

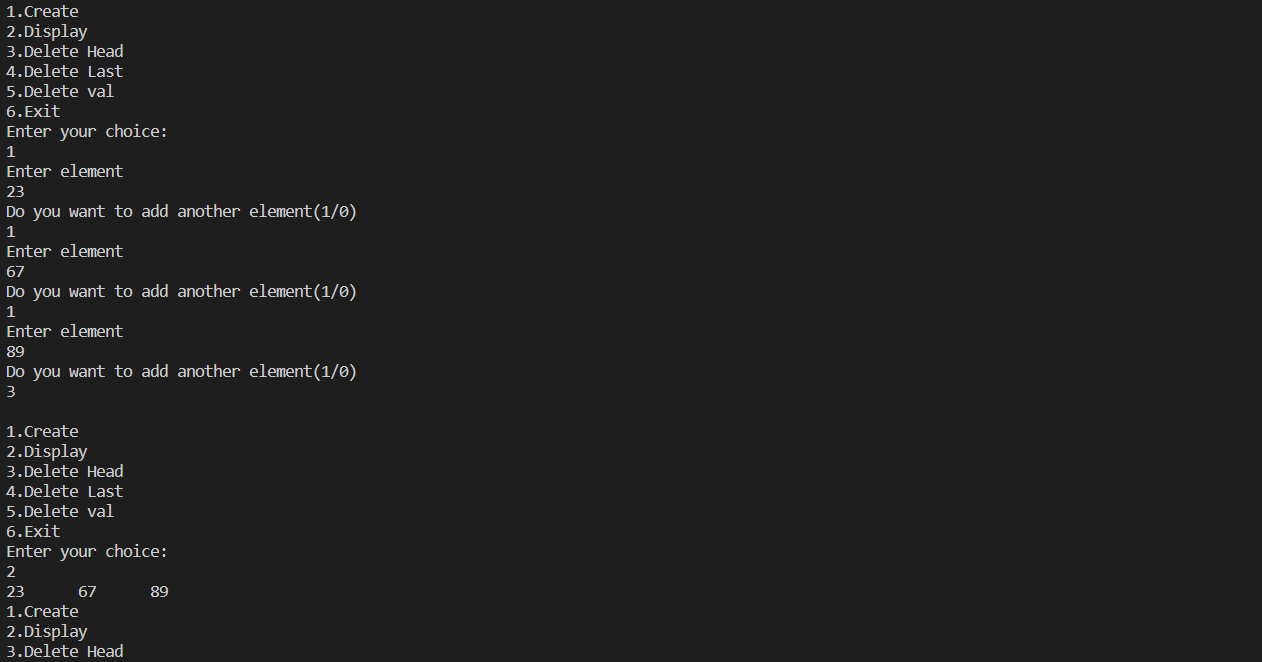
}

prevptr->link = ptr->link;

free(ptr);

}

**OUTPUT :**

****

****

**LAB PROGRAM 7 :**

**Q : WAP to Implement Single Link List with following operations**

**a) Sort the linked list.**

**b) Reverse the linked list.**

**c) Concatenation of two linked lists**

**Code :**

#include <stdio.h>

#include <stdlib.h>

struct NODE

{

int data;

struct NODE \*link;

};

typedef struct NODE Node;

Node \*Start, \*new, \*curr;

void create()

{

int ch;

Start = (Node \*)malloc(sizeof(Node));

printf("enter the element\n");

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

curr = Start;

printf("do you want to enter more elements(1/0)\n");

scanf("%d", &ch);

while (ch == 1)

{

new = (Node \*)malloc(sizeof(Node));

printf("enter the element\n");

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

curr->link = new;

curr = new;

printf("do you want to enter more elements(1/0)\n");

scanf("%d", &ch);

}

curr->link = NULL;

}

void sort()

{

if (Start == NULL)

{

printf("linked list does not exist\n");

return;

}

else

{

int t, cnt = 0;

int arr[100];

Node \*temp1, \*temp2;

temp1 = Start;

while (temp1 != NULL)

{

arr[cnt] = temp1->data;

temp1 = temp1->link;

cnt++;

}

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

{

for (int j = 0; j < cnt - i; j++)

{

if ((arr[j]) < (arr[j + 1]))

{

t = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = t;

}

}

}

temp2 = Start;

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

{

temp2->data = arr[i];

temp2 = temp2->link;

}

return;

}

}

void reverse()

{

if (Start == NULL)

{

printf("linked list does not exist\n");

return;

}

else

{

int t, cnt = 0;

int arr[100];

Node \*temp1, \*temp2;

temp1 = Start;

while (temp1 != NULL)

{

arr[cnt] = temp1->data;

temp1 = temp1->link;

cnt++;

}

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

{

for (int j = 0; j < cnt - 1 - i; j++)

{

t = arr[j + 1];

arr[j + 1] = arr[j];

arr[j] = t;

}

}

temp2 = Start;

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

{

temp2->data = arr[i];

temp2 = temp2->link;

}

return;

}

}

void concat()

{

printf("enter the linked list to concatenate\n");

int ch;

Node\* Start1 = (Node \*)malloc(sizeof(Node));

printf("enter the element\n");

scanf("%d", &Start1->data);

Node\* curr1 = Start1;

printf("do you want to enter more elements(1/0)\n");

scanf("%d", &ch);

while (ch == 1)

{

Node\* new1 = (Node \*)malloc(sizeof(Node));

printf("enter the element\n");

scanf("%d", &new1->data);

curr1->link = new1;

curr1 = new1;

printf("do you want to enter more elements(1/0)\n");

scanf("%d", &ch);

}

curr1->link = NULL;

curr->link=Start1;

printf("Successfully concatenated\n");

return;

}

void display()

{

Node \*temp;

temp = Start;

do

{

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

temp = temp->link;

} while (temp != NULL);

}

int main()

{

int choice;

do

{

printf("1.create\n2.sort\n3.reverse\n4.concatenate\n5.display\n6.exit\n");

scanf("%d", &choice);

switch (choice)

{

case 1:create();

break;

case 2: sort();

break;

case 3:reverse();

break;

case 4: concat();

break;

case 5:display();

break;

case 6:exit(0);

default:printf("enter a valid choice\n");

break;

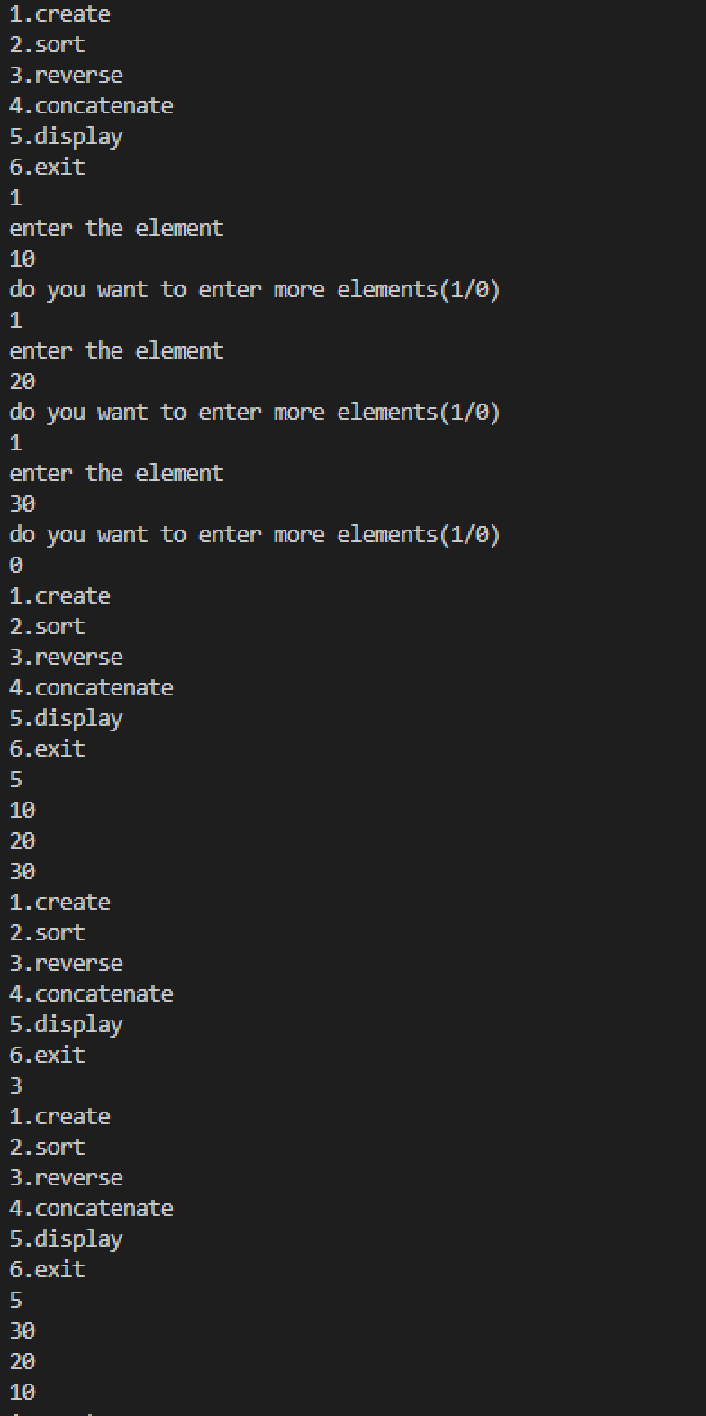
}

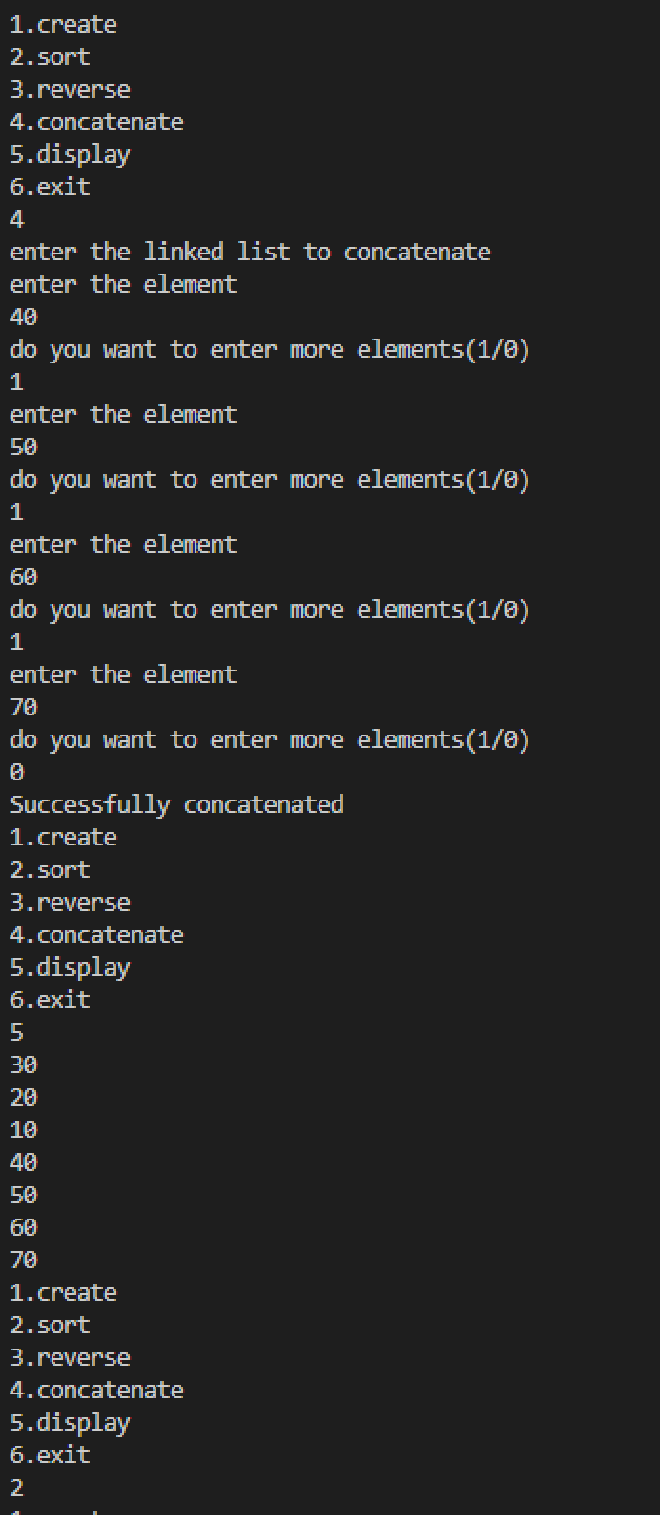
} while (choice != 6);

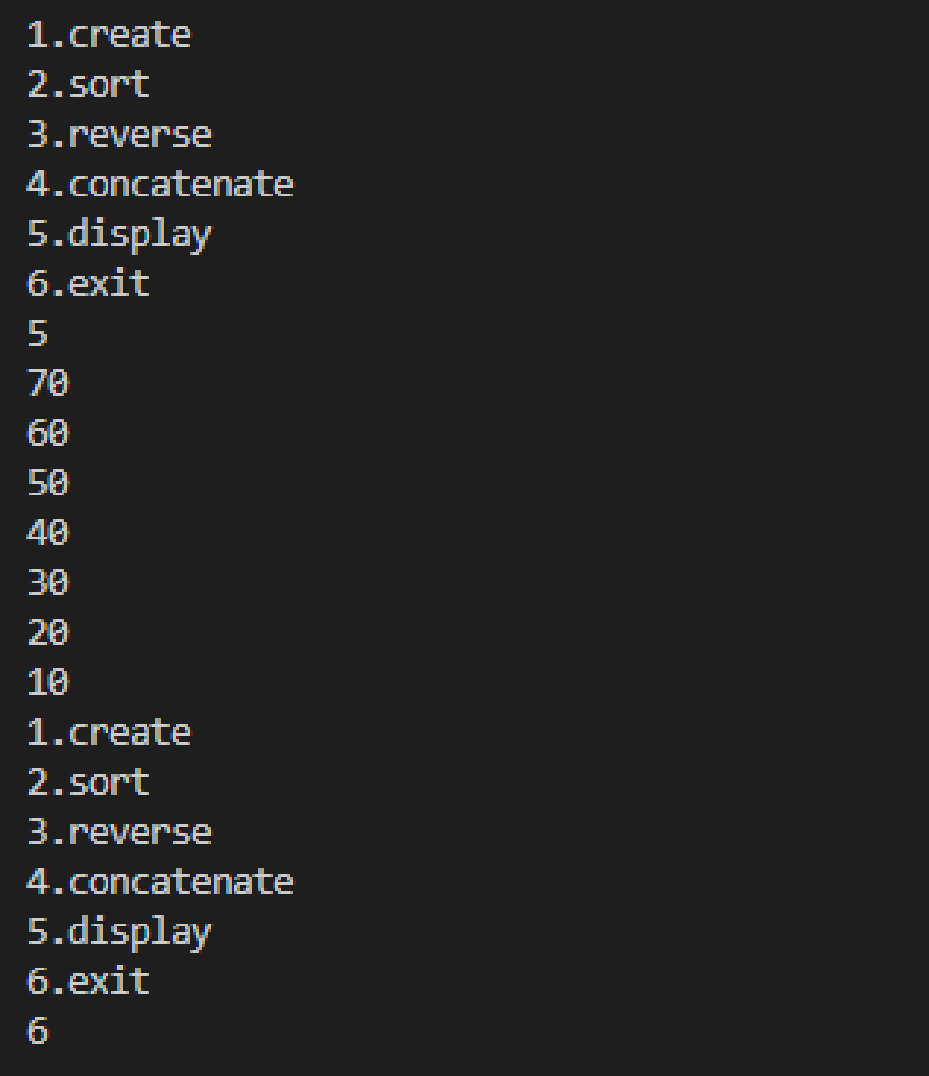
return 0;

}

**OUTPUT :**

****

****

****

**LAB PROGRAM 8 : Implement Stack & Queues using Linked Representation**

**Q : WAP to implement Stack & Queues using Linked Representation**

**Code :**

// PROGRAM TO IMPLEMENT STACKS USING LL

#include <stdio.h>

#include <stdlib.h>

struct NODE

{

int data;

struct NODE \*link;

};

typedef struct NODE Node;

Node \*top, \*new;

void push()

{

if (top == NULL)

{

top = (Node \*)malloc(sizeof(Node));

printf("enter the element\n");

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

top->link=NULL;

return;

}

else

{

new = (Node \*)malloc(sizeof(Node));

new->link = top;

top = new;

printf("enter the element\n");

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

return;

}

}

void pop()

{

if (top == NULL)

{

printf("Stack underflow\n");

return;

}

else

{

Node \*temp = top;

printf("deleted: %d\n", top->data);

top = top->link;

free(temp);

}

}

void display()

{

if (top == NULL)

{

printf("stack underflow\n");

return;

}

else

{

Node \*temp;

temp = top;

do

{

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

temp = temp->link;

} while (temp != NULL);

}

}

int main(){

int choice;

while(1)

{

printf("1.push\n2.pop\n3.display\n4.exit\n");

scanf("%d",&choice);

switch (choice)

{

case 1:push();

break;

case 2: pop();

break;

case 3:display();

break;

case 4:exit(0);

default:printf("enter a valid choice\n");

break;

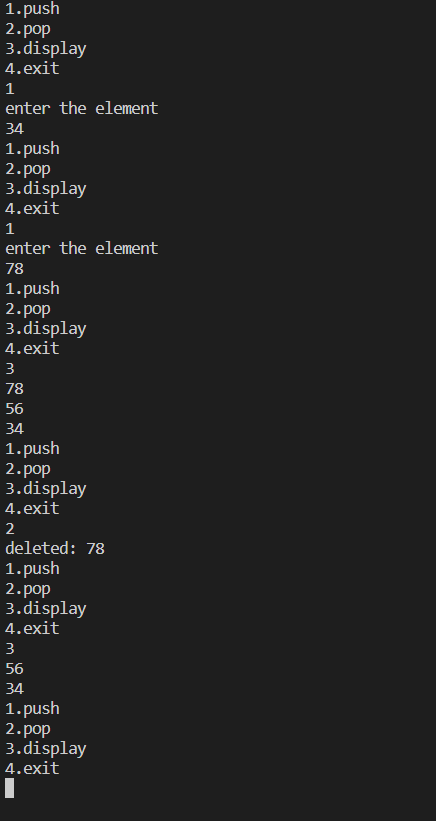
}

}

return 0;

}

**OUTPUT :**

****

**Code :**

// PROGRAM TO IMPLEMENT QUEUE USING LL

#include <stdio.h>

#include <stdlib.h>

struct NODE

{

int data;

struct NODE \*link;

};

typedef struct NODE Node;

Node \*front, \*rear,\*new;

void insert(){

if(rear==NULL){

front=(Node \*)malloc(sizeof(Node));

rear=front;

printf("enter the element\n");

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

return;

}

else{

new=(Node \*)malloc(sizeof(Node));

rear->link=new;

rear=new;

printf("enter the element\n");

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

new->link=NULL;

return;

}

}

void delete(){

if(front==NULL){

printf("empty QUEUE\n");

return;

}

else{

printf("deleted: %d\n",front->data);

Node \*temp;

temp=front;

front=front->link;

free(temp);

}

}

void display(){

Node \*temp;

temp=front;

do{

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

temp=temp->link;

}while(temp!=NULL);

}

int main(){

int choice;

while(1)

{

printf("1.insert\n2.delete\n3.display\n4.exit\n");

scanf("%d",&choice);

switch (choice)

{

case 1:insert();

break;

case 2:delete();

break;

case 3:display();

break;

case 4:exit(0);

default:printf("enter a valid chocie\n");

break;

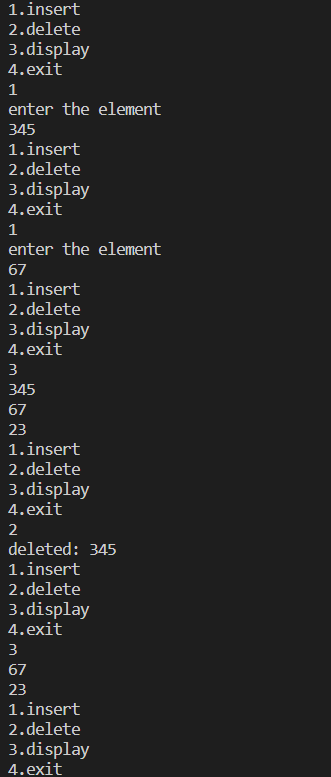
}

}

return 0;

}

**OUTPUT :**



**LAB PROGRAM 9 :**

**Q : WAP to Implement doubly link list with primitive operations**

**a) Create a doubly linked list.**

**b) Insert a new node to the left of the node.**

**c) Delete the node based on a specific value**

**d) Display the contents of the list**

**Code :**

#include <stdio.h>

#include <stdlib.h>

struct node{

struct node\* rlink;

struct node\* llink;

int data;

};

typedef struct node node;

node \*start,\*new,\*curr;

void create(){

int ch;

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

printf("enter the element\n");

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

curr=start;

curr->llink=NULL;

printf("do you want to enter more elements(1/0)\n");

scanf("%d",&ch);

while(ch==1){

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

printf("enter the element\n");

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

curr->rlink=new;

new->llink=curr;

curr=new;

printf("do you want to enter more elements(1/0)\n");

scanf("%d",&ch);

}

curr->rlink=NULL;

}

void insert(){

if(start==NULL){

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

printf("enter the element\n");

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

start->llink=NULL;

start->rlink=NULL;

curr=start;

return;

}

else{

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

printf("enter the element\n");

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

new->rlink=start;

start->llink=new;

start=new;

start->llink=NULL;

return;

}

}

void delete(int del){

if(start==NULL){

printf("empty linked list\n");

return;

}

else{

node \*temp=start;

while(temp->rlink!=NULL && temp->data!=del){

temp=temp->rlink;

}

if(temp->data==del){

if(temp->rlink==NULL){

printf("deleted: %d\n",temp->data);

temp->llink->rlink=NULL;

free(temp);

}

else if(temp==start){

printf("deleted: %d\n",temp->data);

start=start->rlink;

free(temp);

}

else{

printf("deleted: %d\n",temp->data);

temp->llink->rlink=temp->rlink;

temp->rlink->llink=temp->llink;

free(temp);

return;

}

}

else{

printf("element not found\n");

return;

}

}

}

void display(){

node \*temp;

temp=start;

do{

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

temp=temp->rlink;

}while(temp!=NULL);

}

int main(){

int choice, del;

do{

printf("1.create\n2.insert at left\n3.delete element\n4.display\n5.exit\n");

scanf("%d",&choice);

switch (choice)

{

case 1:create();

break;

case 2:insert();

break;

case 3:printf("enter the element to delete\n");

scanf("%d",&del);

delete(del);

break;

case 4:display();

break;

case 5: exit(0);

default:printf("enter a valid chocie\n");

break;

}

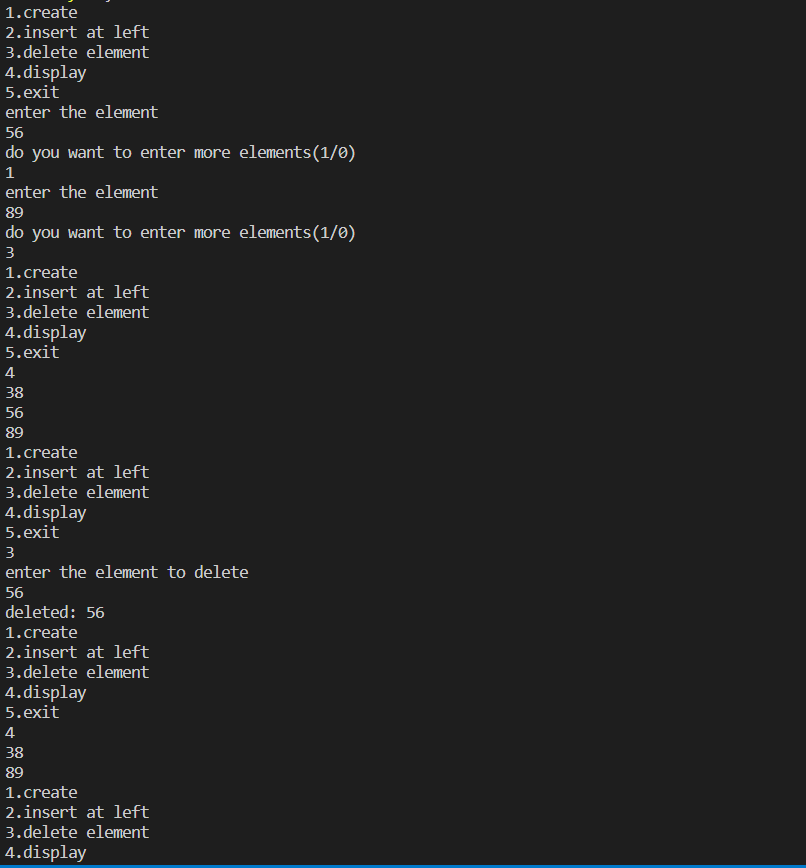
}

while(choice!=5);

return 0;

}

**OUTPUT :**

****

**LAB PROGRAM 10 :**

**Q : Write a program  
a) To construct a binary Search tree.  
b) To traverse the tree using all the methods i.e., in-order, preorder and postorder  
c) To display the elements in the tree.**

**Code :**

#include <stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node \*left;

struct Node \*right;

};

struct Node \*getNode()

{

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

if(temp==NULL)

{

printf("Memory full!\n");

return NULL;

}

else

{

return temp;

}

}

void inorder(struct Node \*root)

{

if(root!=NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

void preorder(struct Node \*root)

{

if(root!=NULL)

{

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

preorder(root->left);

preorder(root->right);

}

}

void postorder(struct Node \*root)

{

if(root!=NULL)

{

postorder(root->left);

postorder(root->right);

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

}

}

struct Node \*createBST(struct Node \*root, int data)

{

struct Node \*new=getNode();

new->data=data;

new->right=NULL;

new->left=NULL;

if(root==NULL)

{

root=new;

return root;

}

else

{

if(data<root->data)

{

if(root->left==NULL)

{

root->left=new;

}

else

{

root->left=createBST(root->left,data);

}

}

else if(data>root->data)

{

if(root->right==NULL)

{

root->right=new;

}

else

{

root->right=createBST(root->right,data);

}

}

return root;

}

}

void main()

{

struct Node \*root=NULL;

int n,data;

while(1){

printf("Enter 1 to create a binary search tree.\nEnter 2 to get preorder traversal.\nEnter 3 to get inorder traversal.\n");

printf("Enter 4 to get postorder traversal.\nEnter 5 to exit!\n");

scanf("%d",&n);

switch(n)

{

case 1:

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

scanf("%d",&data);

root=createBST(root,data);

break;

case 2:

preorder(root);

printf("\n");

break;

case 3:

inorder(root);

printf("\n");

break;

case 4:

postorder(root);

printf("\n");

break;

case 5:

exit(0);

default:

printf("Invalid choice!\n");

}

}

}

**OUTPUT:**