**VADODARA INSTITUTE OF ENGINEERING**

**KOTAMBI**

**Lab Manual**

****

**Data Structures (DS)**

**(Subject Code: 3130702)**

**(III Semester CE/IT)**

**Prepared By:**

CE/IT Department

**Practical List**

|  |  |
| --- | --- |
| 1 | Introduction to pointers. Call by Value and Call by reference. |
| 2 | Introduction to Dynamic Memory Allocation. DMA functions malloc(), calloc(), free() etc. |
| 3 | Implement a program for stack that performs following operation using array. a) PUSH b) POP |
| 4 | Implement a program for stack that performs following operation using array. PEEP b) CHANGE c) DISPLAY |
| 5 | Implement a program to convert infix notation to postfix notation using stack. |
| 6 | Write a program to implement simple queue using arrays that performs following operations (a) INSERT (b) DELETE (c) DISPLAY |
| 7 | Write a program to implement Circular Queue using arrays that performs following Operations. (a) INSERT (b) DELETE (c) DISPLAY |
| 8 | Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the front of the linked list. |
| 9 | Write a menu driven program to implement following operation on the singly linked list. a) Insert a node at the end of the linked list. |
| 10 | Write a menu driven program to implement following operation on the singly linked list. a) Delete a first node of the linked list. |
| 11 | Write a menu driven program to implement following operation on the singly linked list. a) Delete a node before specified position. |
| 12 | Write a menu driven program to implement following operation on the singly linked list. a) Delete a node after specified position |
| 13 | Write a program to implement stack using linked list. |
| 14 | Write a program to implement queue using linked list. |
| 15 | Write a program to implement following operations on the doubly linked list. a) Insert a node at the front of the linked list. |
| 16 | Write a program to implement following operations on the doubly linked list. a) Insert a node at the end of the linked list. |
| 17 | Write a program to implement following operations on the doubly linked list. a) Delete a last node of the linked list. |
| 18 | Write a program to implement following operations on the doubly linked list. a) Delete a node after specified position. |
| 19 | Write a program to implement following operations on the circular linked list. a) Insert a node at the end of the linked list. |
| 20 | Write a program to implement following operations on the circular linked list. a) Insert a node at specified position. |
| 21 | Write a program to implement following operations on the circular linked list. a) Delete a first node. |

|  |  |
| --- | --- |
| 22 | Write a program to implement following operations on the circular linked list. a) Delete the last node. |
| 23 | Implement recursive or non-recursive tree traversing methods of Inorder traversal. |
| 24 | Implement recursive or non-recursive tree traversing methods of Preorder traversal. |
| 25 | Implement recursive or non-recursive tree traversing methods of Postorder traversal. |
| 26 | Write a program to implement Merge Sort |
| 27 | Write a program to implement Bubble Sort |
| 28 | Write a program to implement Selection Sort |

**Practical – 1**

**AIM:** Introduction to pointers. Call by Value and Call by reference.

**PROGRAM:**

include<stdio.h>

#include<conio.h>

void swap(int\*num1,int\*num2)

{

int temp ;

temp =\*num1 ;

\*num1 =\*num2 ;

\*num2 = temp ;

}

void swapp(int num1,int num2)

{

int temp;

temp=num1;

num1=num2;

num2=temp;

}

void main()

{

int num1,num2;

clrscr();

printf("\nEnter two numbers no.1 and no.2 : ");

scanf("%d %d",&num1,&num2);

printf("\nbefore swapping");

printf("\nNo. 1 : %d",num1);

printf("\nNo. 2 : %d",num2);

printf(" \nafter swapping, CALL BY VALUE");

swapp(num1,num2);

printf("\nafter swapping");

printf("\nNo. 1 : %d",num1);

printf("\nNo. 2 : %d",num2);

printf(" \nafter swapping, CALL BYREFERENCE");

swap(&num1,&num2);

printf("\nafter swapping");

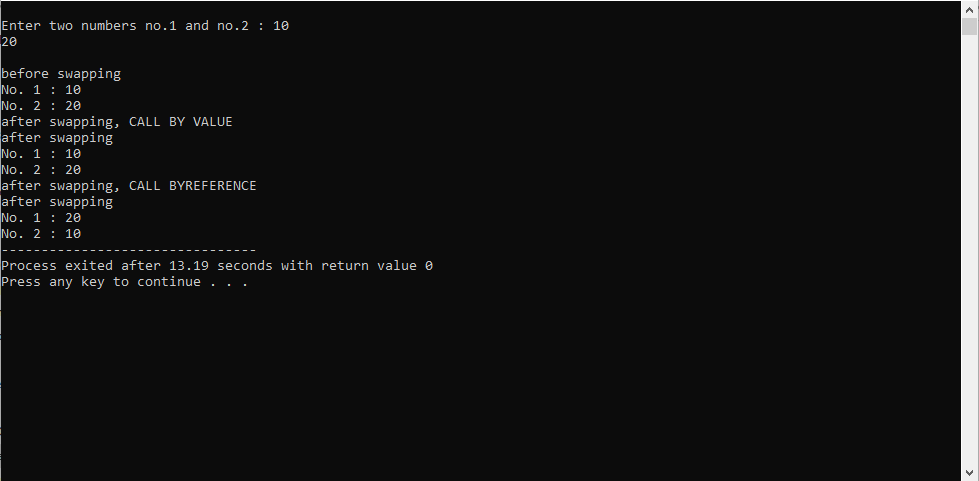
printf("\nNo. 1 : %d",num1);

printf("\nNo. 2 : %d",num2);

getch();

}

**OUTPUT:**

****

**Practical – 2**

**AIM:** Introduction to Dynamic Memory Allocation. DMA functions malloc(),

calloc(), free() etc.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<malloc.h>

void main()

{

Char \*ma;

Clrscr();

//allocating memory space.

ma = malloc(sizeof(char));

ma=”hello”;

strcat(ma,”VIE,”);

printf(“dynamically allocated value: %s\n”ma);

// reallocating memory space.

ma = reallocate(ma,100\*sizeof(char));

strcat(ma,“kotambi,vadodara”);

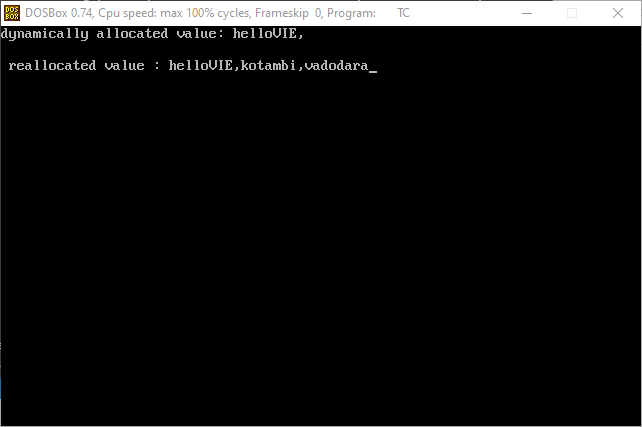
printf(“\n reallocated value : %s”,ma);

free();

getch();

}

**OUTPUT:**

****

**Practical – 3**

**AIM:** To perform PUSH and POP operations on Stack.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

//stdlib for exit function

#include<conio.h>

#define MAX 10

int top=-1,stack[MAX];

void push();

void pop();

void display();

void main()

{

int ch;

while(1)

{

printf("\n\*\*\* Stack Menu \*\*\*"); printf("\n\n1.Push\n2.Pop\n3.display\n4.exit"); printf("\n\nEnter your choice(1-4):"); scanf("%d",&ch);

switch(ch)

{

case 1: push();

break;

case 2: pop();

break

case 3: display();

break;

case 4: exit(0);

break;

default: printf("\nWrong Choice!!");

}

}

}

void push()

{

int val;

if(top==MAX-1)

{

printf("\nStack is full!!");

}

else

{

printf("\nEnter element to push:");

scanf("%d",&val);

top=top+1; //top=0

stack[top]=val;//stack[0]=10

}

}

void pop()

{

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

printf("\nDeleted element is %d",stack[top]); top=top-1;

}

}

void display()

{

int i;

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

printf("\nStack is...\n");

for(i=top;i>=0;--i)

printf("%d\n",stack[i]);

}

}

**OUTPUT:**

****

**Practical – 4**

**AIM:** To perform PEEP and CHANGE operations on Stack

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#define MAX 5

int top=-1,stack[MAX],temp[MAX], i=-1;

void push();

void pop();

void peep();

void change();

void display();

void push()

{

int val;

if(top==MAX-1)

{

printf("\nStack is full!!");

}

else

{

printf("\nEnter element to push:");

scanf("%d",&val);

top=top+1; stack[top]=val;

}

}

void pop()

{

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

printf("\nDeleted element is %d",stack[top]); top=top-1;

}

}

void display()

{

int i;

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{

printf("\nStack is...\n");

for(i=top;i>=0;--i)

printf("%d\n",stack[i]);

}

}

// Peep operation....

void peep(){

printf("\n\tTop : %d", top);//3

printf("\n\tValue: %d",stack[top]);//stack[3]=40

}

void change(int i, int new\_element){

stack[top-i+1] = new\_element;

}

void main()

{

int ch;

int item, row, new\_element;

clrscr();

while(1)

{

printf("\n\*\*\* Stack Menu \*\*\*"); printf("\n\n1.Push\n2.Pop\n3.display\n4.peep\n5.change\n6.exit"); printf("\n\nEnter your choice(1-4):"); scanf("%d",&ch);

switch(ch)

{

case 1: push();

break;

case 2: pop();

break;

case 3: display();

break;

case 4:

peep();

break;

case 5:

printf("\n\tEnter row no : ");

scanf("%d",&row);

printf("\n\tEnter new element: ");

scanf("%d", &new\_element);

change(row, new\_element );

break;

case 6: exit(0);

break;

default: printf("\nWrong Choice!!");

}

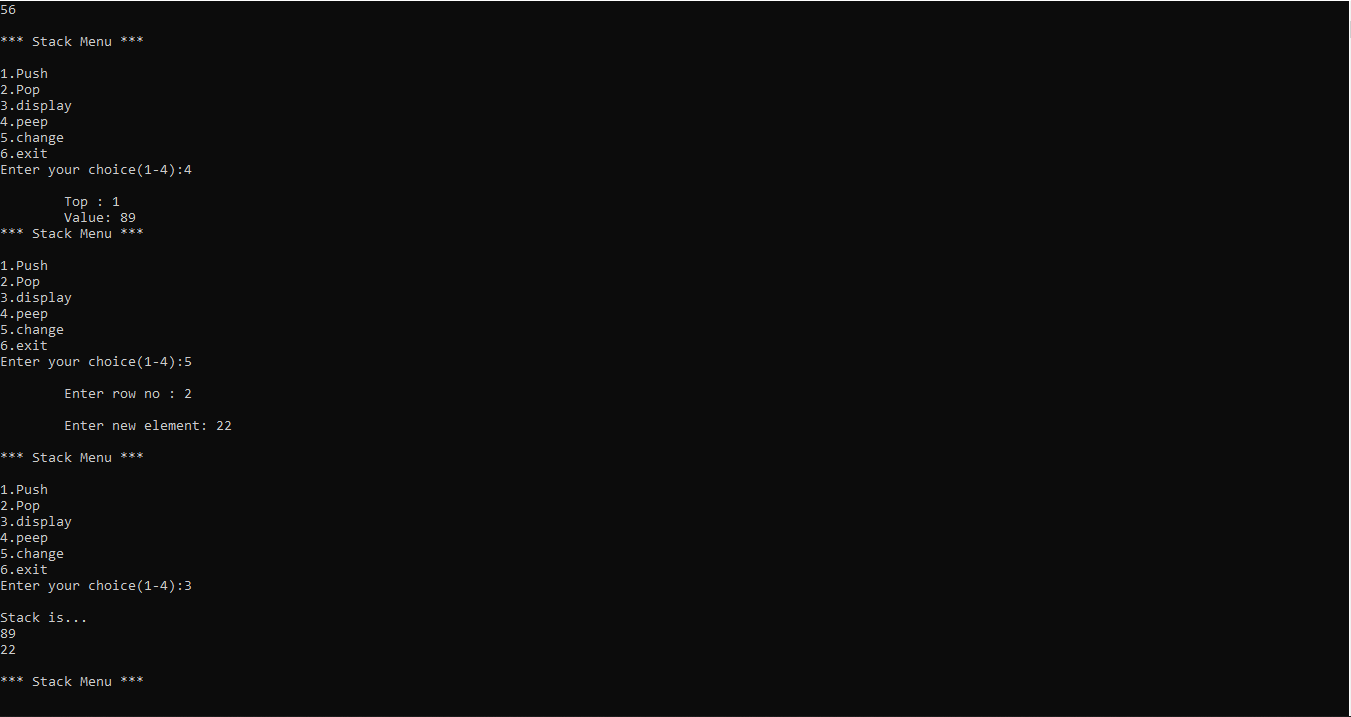
}

getch();

}

**OUTPUT:**

****

****

**Practical – 5**

**AIM:** Implement a program to convert infix notation to postfix notation using

stack.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<ctype.h>//support isalnum() i.e for alpha numeric character

#define MAX 50

typedef struct stack

{

int data[MAX];

int top;

}stack;

int precedence(char);

void init(stack \*);

int empty(stack \*);

int full(stack \*);

int pop(stack \*);

void push(stack \*,int);

int top(stack \*); //value of the top element

void infix\_to\_postfix(char infix[],char postfix[]);

void main()

{

char infix[30],postfix[30];

clrscr();

printf("Enter an infix expression(eg: 5+2\*4): ");

gets(infix);

infix\_to\_postfix(infix,postfix);

printf("\nPostfix expression: %s",postfix);

getch();

}

void infix\_to\_postfix(char infix[],char postfix[])

{

stack s;

char x,token;

int i,j; //i-index of infix,j-index of postfix

// init(&s);

s.top=-1;

j=0;

for(i=0;infix[i]!='\0';i++)

{

token=infix[i];

if(isalnum(token))

postfix[j++]=token;

else

if(token=='(')

push(&s,'(');

else

if(token==')')

while((x=pop(&s))!='(')

postfix[j++]=x;

else

{

while(precedence(token)<=precedence(top(&s))&&!empty(&s))

{

x=pop(&s);

postfix[j++]=x;

}

push(&s,token);

}

}

while(!empty(&s))

{

x=pop(&s);

postfix[j++]=x;

}

postfix[j]='\0';

}

int precedence(char x)

{

if(x=='(')

return(0);

if(x=='+'||x=='-')

return(1);

if(x=='\*'||x=='/'||x=='%')

return(2);

return(0);

}

//void init(stack \*s)

//{

// s->top=-1;

//}

int empty(stack \*s)

{

if(s->top==-1)

return(1);

return(0);

}

int full(stack \*s)

{

if(s->top==MAX-1)

return(1);

return(0);

}

void push(stack \*s,int x)

{

s->top=s->top+1;

s->data[s->top]=x;

}

int pop(stack \*s)

{

int x;

x=s->data[s->top];

s->top=s->top-1;

return(x);

}

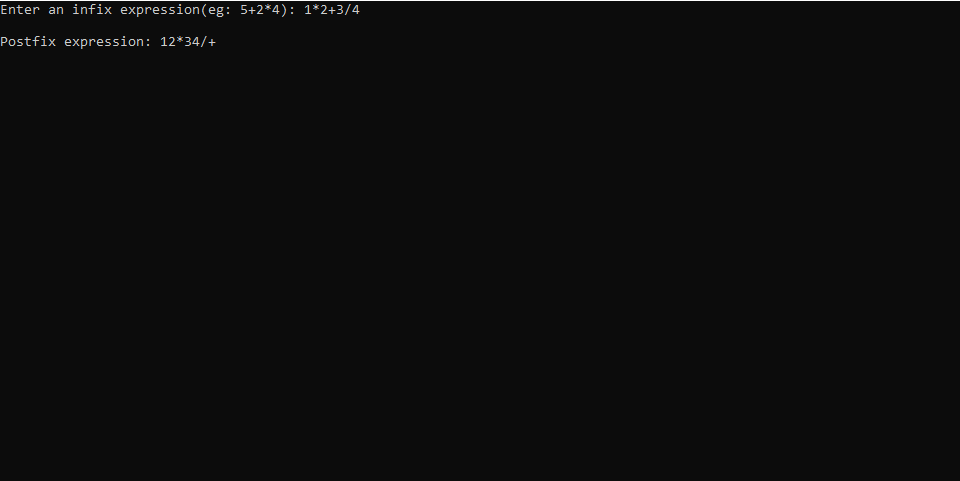
int top(stack \*p)

{

return (p->data[p->top]);

}

**RESULT:**

****

**Practical – 6**

**AIM:** To implement simple queue using array and perform INSERT, DELETE and DISPLAY operations

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define max 10

int q[10],front=-1,rear=-1;

void insert();

void delet();

void display();

void main()

{

int ch;

clrscr();

printf("\nQueue operations\n");

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

while(1)

{

printf("Enter your choice:"); scanf("%d",&ch);

switch(ch)

{

case 1:insert();

break;

case 2:delet();

break;

case 3:display();

break;

case 4:exit(0);

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

}

}

getch();

}

void insert()

{

int x;

if(rear==max-1)

printf("Queue is overflow\n");

else

{

if(front == -1)

front=0;

printf("Enter element to be insert:"); scanf("%d",&x);

rear=rear+1;

q[rear]=x;

}

}

void delet()

{

int a;

if((front==-1)&&(rear==-1))

{

printf("Queue is underflow\n");

}

a=q[front];

front=front+1;

printf("Deleted element is:%d\n",a);

if(front>rear)

{

front=-1; rear=-1;//queue is empty

}

}

void display()

{

int i;

if(front==-1 && rear==-1)

{

printf("Queue is underflow\n");

}

for(i=front;i<=rear;i++)

{

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

printf("\n");

}

getch();

}

**OUTPUT:**

****

**Practical – 7**

**AIM:** To implement circular queue using array and perform INSERT, DELETE and DISPLAY operations.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define max 10

int q[10],front=-1,rear=-1;

void insert();

void delet();

void display();

void main()

{

int ch;

clrscr();

printf("\nQueue operations\n");

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

while(1)

{

printf("Enter your choice:"); scanf("%d",&ch);

switch(ch)

{

case 1:insert();

break;

case 2:delet();

break;

case 3:display();

break;

case 4:exit(0);

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

}

}

getch();

}

void insert()

{

int x;

if(rear==max-1)

printf("Queue is overflow\n");

else

{

if(front == -1)

front=0;

printf("Enter element to be insert:"); scanf("%d",&x);

rear=rear+1;

q[rear]=x;

}

}

void delet()

{

int a;

if((front==-1)&&(rear==-1))

{

printf("Queue is underflow\n");

}

a=q[front];

front=front+1;

printf("Deleted element is:%d\n",a);

if(front>rear)

{

front=-1; rear=-1;//queue is empty

}

}

void display()

{

int i;

if(front==-1 && rear==-1)

{

printf("Queue is underflow\n");

}

for(i=front;i<=rear;i++)

{

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

printf("\n");

}

getch();

}

**OUTPUT:**

****

**PRACTICAL – 8**

**AIM:** Write a menu driven program to implement following operation on the singly linked list. a) Insert node at start of the linked list.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

void insert\_beg();

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*start=NULL;

int main()

{

int ch;

for(;;)//infinity loop

{

printf("\n \*\*\*LINKLIST MENU\*\*\*");

printf("\n\n1.insert\_beg\n2.display\n3.exit");

printf("\n\n enter your choice (1 2 or 3)- ");

scanf("%d",&ch);

switch(ch)

{

case 1:insert\_beg();

break;

case 2:display();

break;

case 3:exit(0);

default:printf("\nwrong coice!");

break;

}

}

}

void insert\_beg()

{

struct node \*new\_node;

int val;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

new\_node->next=start;

start=new\_node;

}

void display()

{

struct node \*ptr;

ptr=start;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

**OUTPUT:**

****

**Practical – 9**

**AIM:** Write a menu driven program to implement following operation on the singly linked list. a) Insert node at end of the linked list.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node \* insert\_beg();

struct node \* insert\_end();

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*start=NULL;

int main()

{

int ch;

while(1)

{

printf("\n \*\*\*LINKLIST MENU\*\*\*");

printf("\n\n1. Insert\_beg\n2. Insert\_end\n3. Display\n4. Exit");

printf("\n\n Enter your choice (1 2 3 or 4)- ");

scanf("%d",&ch);

switch(ch)

{

case 1:start=insert\_beg(); break;

case 2:start=insert\_end(); break;

case 3:display(); break;

case 4:exit(0);

break;

default:printf("\nwrong coice!");

break;

}

}

}

struct node \* insert\_beg()

{

struct node \*new\_node;

int val;

new\_node=(struct node\*)(malloc(sizeof(struct node))); printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val; new\_node->next=start; start=new\_node;

return start;

}

struct node \* insert\_end()

{

struct node \*new\_node,\*ptr;

int val,i=1;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

new\_node->next=NULL;

ptr=start;

if(start==NULL) //if link list is empty

{

start=new\_node;

}

else

{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

return start;

}

void display()

{

struct node \*ptr;

ptr=start;

while(ptr!=NULL)

{

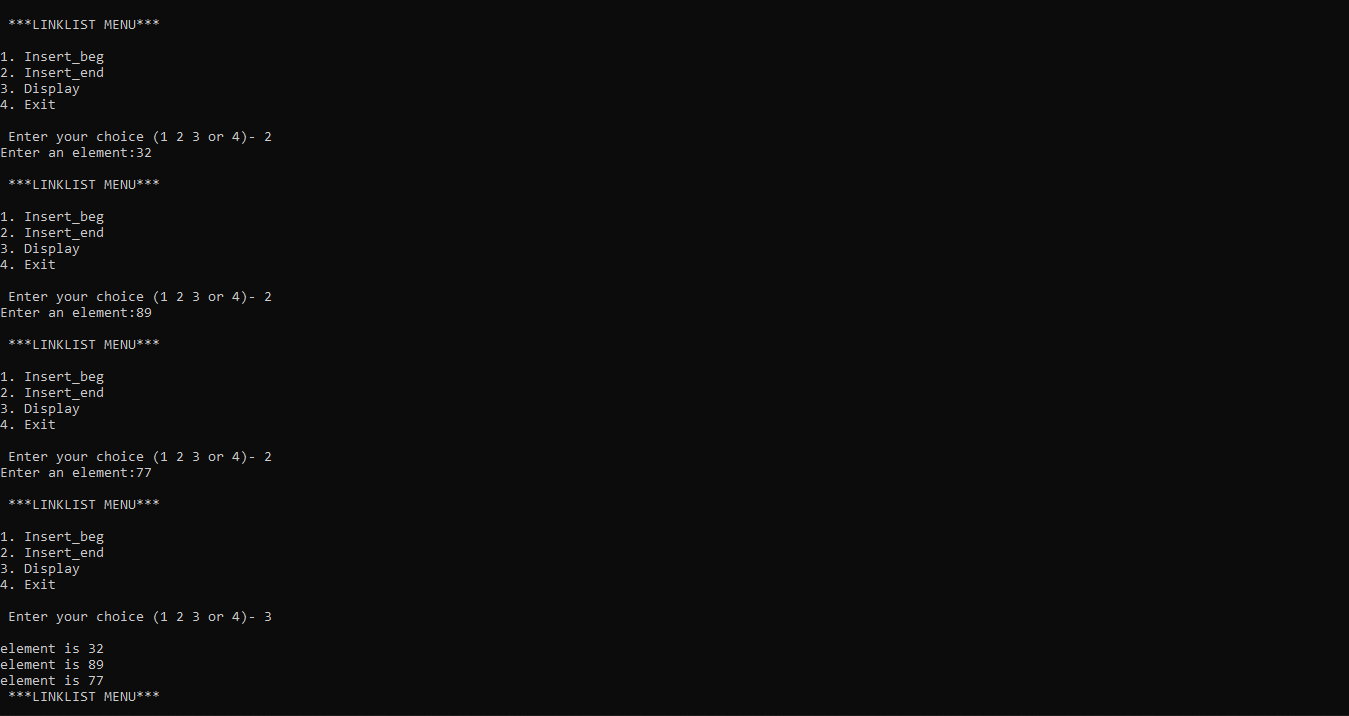
printf("\nelement is %d",ptr->data);

ptr=ptr->next;

}

}

**OUTPUT:**

****

**Practical – 10**

**AIM:** Write a menu driven program to implement following operation on the singly linked list. a) Delete first node of the linked list.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*start=NULL,\*q,\*new\_node;

// struct node \*start=null;

int main()

{

int ch;

void insert\_beg();

//void insert\_end();

void display();

void delete\_beg();

while(1)

{

printf("\n\n---- Singly Linked List(SLL) Menu ----");

printf("\n1.Insert at beginning\n2.Delete at beginning\n3.Display\n4.Exit\n\n");

printf("Enter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: insert\_beg();

break;

case 2: delete\_beg();

break;

case 3: display();

break;

case 4: exit(0);

break;

default: printf("Wrong Choice!!");

}

}

return 0;

}

void insert\_beg()

{

int num;

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

printf("Enter data:");

scanf("%d",&num);

new\_node->data=num;

if(start==NULL) //If list is empty

{

new\_node->next=NULL;

start=new\_node;

}

else

{

new\_node->next=start;

start=new\_node;

}

}

void delete\_beg()

{

if(start==NULL)

{

printf("The list is empty!!");

}

else

{

q=start;

start=start->next;

printf("Deleted element is %d",q->data);

free(q);

}

}

void display()

{

if(start==NULL)

{

printf("List is empty!!");

}

else

{

q=start;

printf("The linked list is:\n");

while(q!=NULL)

{

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

q=q->next;

}

}

}

**OUTPUT:**

****

**Practical – 11**

**AIM:** Write a menu driven program to implement following operation on the singly linked list. a) Delete node before given node of the linked list.

**PROGRAM:**

#include<stdio.h>

//#include<process.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*start=NULL,\*ptr,\*new\_node;

int main()

{

int ch;

void insert\_end();

void display();

void del\_before();

while(1)

{

printf("\n\n---- Singly Linked List(SLL) Menu ----");

printf("\n1.Insert at end\n2.Delete node before specific node\n3.Display\n4.Exit\n\n\t");

printf("Enter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: insert\_end();

break;

case 2: del\_before();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("Wrong Choice!!");

}

}

}

void insert\_end()

{

int num;

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

printf("\tEnter data:");

scanf("%d",&num);

new\_node->data=num;

new\_node->next=NULL;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=new\_node;

}

}

void display()

{

if(start==NULL)

{

printf("List is empty!!");

}

else

{

ptr=start;

printf("\tThe linked list is:\n\t");

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void del\_before()

{

int info;

printf("Enter node info before you want to delete:");

scanf("%d",&info);

struct node \*t,\*t2,\*t3;

t=start;

if(info==start->data)

{

printf("\tNODE CANNOT BE DELETED\n");

}

else

{

if(info==start->next->data)

{

t3=start;

start=start->next;

free(t3);

}

else

{

while(t->next->next->data!=info && t->next->next!=NULL)

{

t=t->next;

}

if(t->next->next->data==info)

{

t2=t->next;

t->next=t2->next;

free(t2);

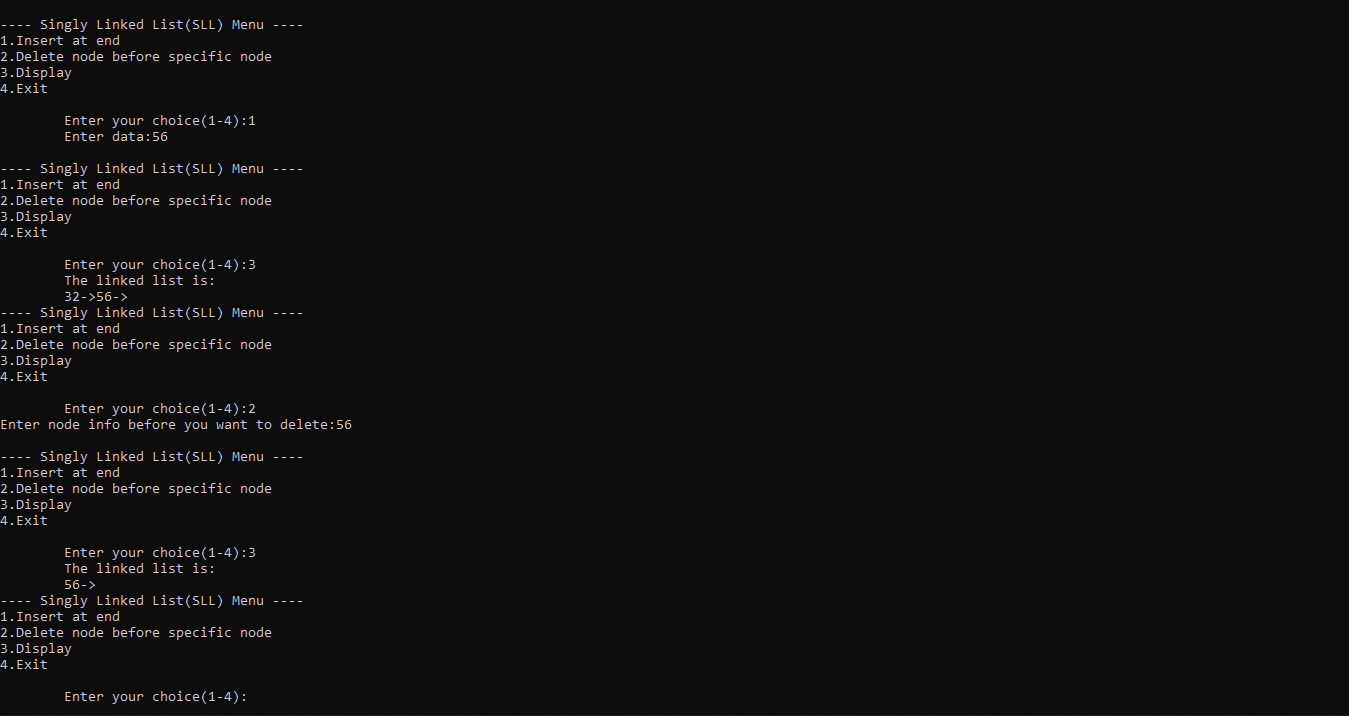
}

}

}

}

**OUTPUT:**

****

**Practical – 12**

**AIM:** Write a menu driven program to implement following operation on the singly linked list. a) Delete node after given node of the linked list.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

//#include<process.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*start=NULL,\*ptr,\*new\_node;

int main()

{

int ch;

void insert\_end();

void display();

void del\_after();

while(1)

{

printf("\n\n---- Singly Linked List(SLL) Menu ----");

printf("\n1.Insert at end\n2.Delete node after specific node\n3.Display\n4.Exit\n\n");

printf("Enter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: insert\_end();

break;

case 2: del\_after();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("Wrong Choice!!");

}

}

}

void insert\_end()

{

int num;

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

printf("Enter data:");

scanf("%d",&num);

new\_node->data=num;

new\_node->next=NULL;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=NULL)

ptr=ptr->next;

ptr->next=new\_node;

}

}

void display()

{

if(start==NULL)

{

printf("List is empty!!");

}

else

{

ptr=start;

printf("The linked list is:\n");

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

}

void del\_after()

{

int info;

printf("Enter node info after you want to delete:");

scanf("%d",&info);

struct node \*t,\*ptr,\*t1;

ptr=start;

if(info==start->data)

{

t=start->next;

start->next=t->next;

free(t);

}

while(ptr->next!=NULL)

{

ptr=ptr->next;

if(ptr->data==info)

{

t1=ptr->next;

if(t1->next==NULL)

{

ptr->next=NULL;

}

else

{

ptr->next=t1->next;

}

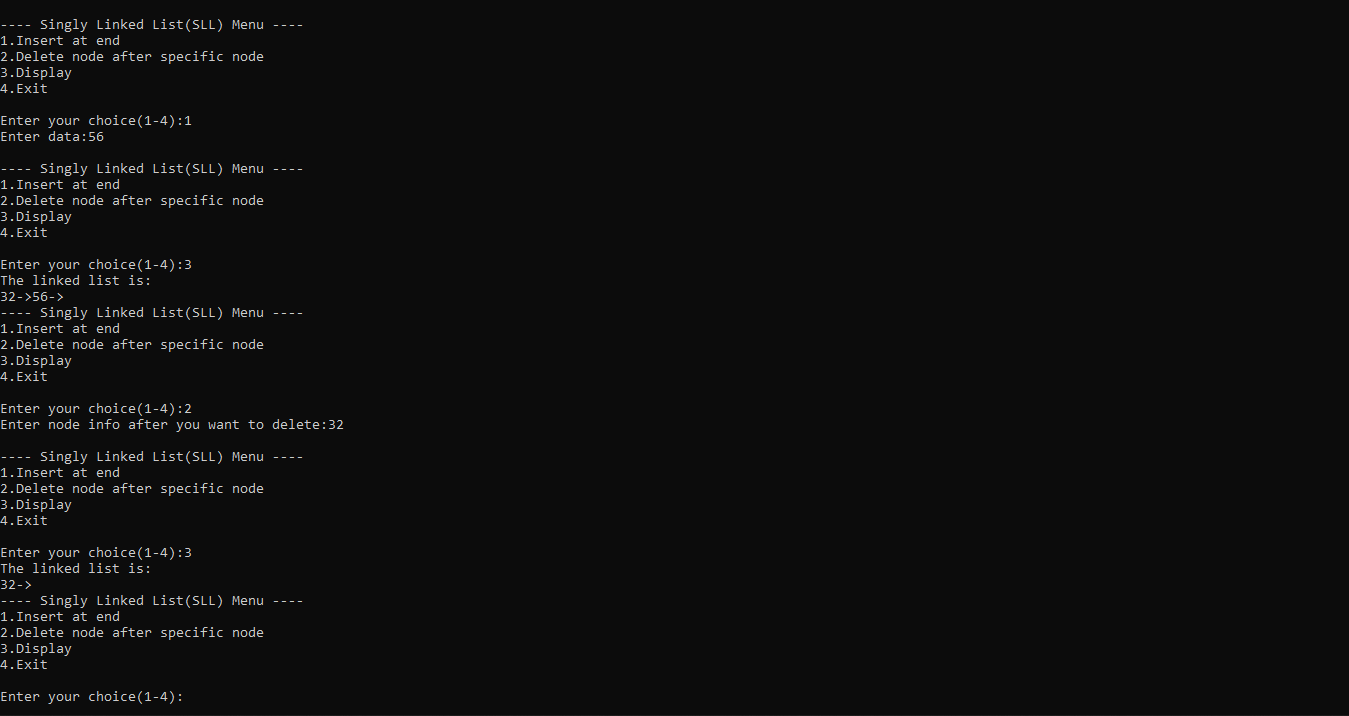
free(t1);

}

}

}

**OUTPUT:**

****

**Practical – 13**

**AIM:** Write a program to implement stack using linked list.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int info;

struct node \*ptr;

}\*top=NULL,\*top1,\*temp;

void push(int data);

void pop();

void display();

void main()

{

int no, ch, e;

printf("\n 1 - Push");

printf("\n 2 - Pop");

printf("\n 3 - Dipslay");

printf("\n 4 - Exit");

while (1)

{

printf("\n Enter choice : "); scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter data : ");

scanf("%d", &no); push(no);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default :

printf(" Wrong choice, Please enter correct choice "); break;

}

}

}

/\* Push data into stack \*/

void push(int data)

{

if (top == NULL)

{

top =(struct node \*)malloc(1\*sizeof(struct node));

top->ptr = NULL;

top->info = data;

}

else

{

temp =(struct node \*)malloc(1\*sizeof(struct node));

temp->ptr = top;

temp->info = data;

top = temp;

}

}

/\* Display stack elements \*/

void display()

{

top1 = top;

if (top1 == NULL)

{

printf("Stack is empty"); return;

}

while (top1 != NULL)

{

printf("%d ", top1->info); top1 = top1->ptr;

}

}

/\* Pop Operation on stack \*/

void pop()

{

top1 = top;

if (top1 == NULL)

{

printf("\n Error : Trying to pop from empty stack"); return;

}

else

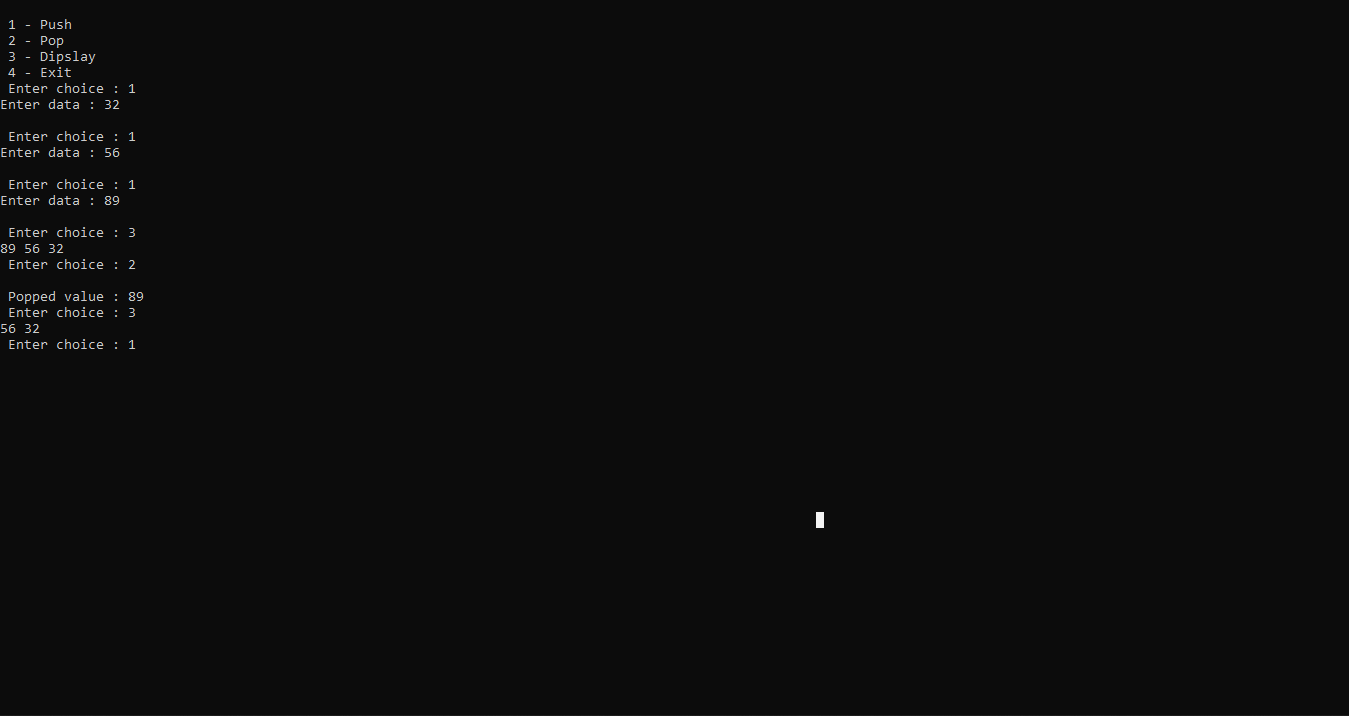
top1 = top1->ptr;

printf("\n Popped value : %d", top->info);

free(top); top = top1;

}

**OUTPUT:**

****

**Practical – 14**

**AIM:** Write a program to implement queue using linked list.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

}\*f=NULL,\*r=NULL,\*ptr,\*newnode;

int ele,a;

void insert();

void delete1();

void display();

int main()

{

int x;

printf("-------QUEUE Menu ");

printf("\n1.insert \n2.delete \n3.display \n4.exit");

while(1){

printf("\nenter your choice ");

scanf("%d",&x);

switch(x)

{

case 1: insert();break;

case 2: delete1();break;

case 3: display();break;

case 4: exit(0);break;

default :

printf(" Wrong choice, Please enter correct choice "); break;

}

}

}

void insert()

{

printf("enter the element ");

scanf("%d",&ele);

newnode=(struct node\*)malloc(sizeof (struct node)); newnode->data=ele;

newnode->next=NULL;

if(r==NULL)

{

r=newnode; f=r;

}

else

{

r->next=newnode;

r=newnode;

}

}

void display()

{

if(f==NULL)

{

printf("link list is empty");

}

else

{

ptr=f;

while(ptr->next!=NULL)

{

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

ptr=ptr->next;

}

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

}

}

void delete1()

{

if(f==NULL)

{

printf("linklist is overflow");

}

else

{

ptr=f;

f=f->next;

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

free (ptr);

}

}

**OUTPUT:**

****

**Practical – 15**

**AIM:** Write a menu driven program to implement following operation on the doubly linked list. a) Insert a node at the front of the doubly linked list.

**PROGRAM:**

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

struct node{

int num;

struct node \*next;

struct node \*prev;

};

struct node \*head=NULL,\*temp, \*first, \*last;

int info;

void display();

void insert\_at\_begin();

int main()

{

int i;

printf("\nprogram for insertion in a doubly linked list :\n");

do {

printf("\n1.Insert element at the begin of the linkedlist :");

printf("\n2.display"); printf("\n3.Exit\n");

printf("\nEnter your choice : ");

scanf("%d",&i);

switch(i) {

case 1: insert\_at\_begin();

break;

case 2:

display();

break; case 3: exit(0);

}

} while(1);

}

void display() {

struct node \*ptr; ptr=head;

printf("\nStatus of the doubly linked list is as follows :\n");

while(ptr!=NULL) /\* traversing the linked list \*/

{ printf("\n%d",ptr->num); ptr=ptr->next; }

}

void insert\_at\_begin() {

printf("\nEnter the value which do you want to insert at begining\n");

scanf("%d",&info);

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

//(struct node)malloc(sizeof(NODE));

temp->num=info; temp->next=NULL;

temp->prev=NULL;

if(head==NULL) { head=temp; last=temp; }

else {

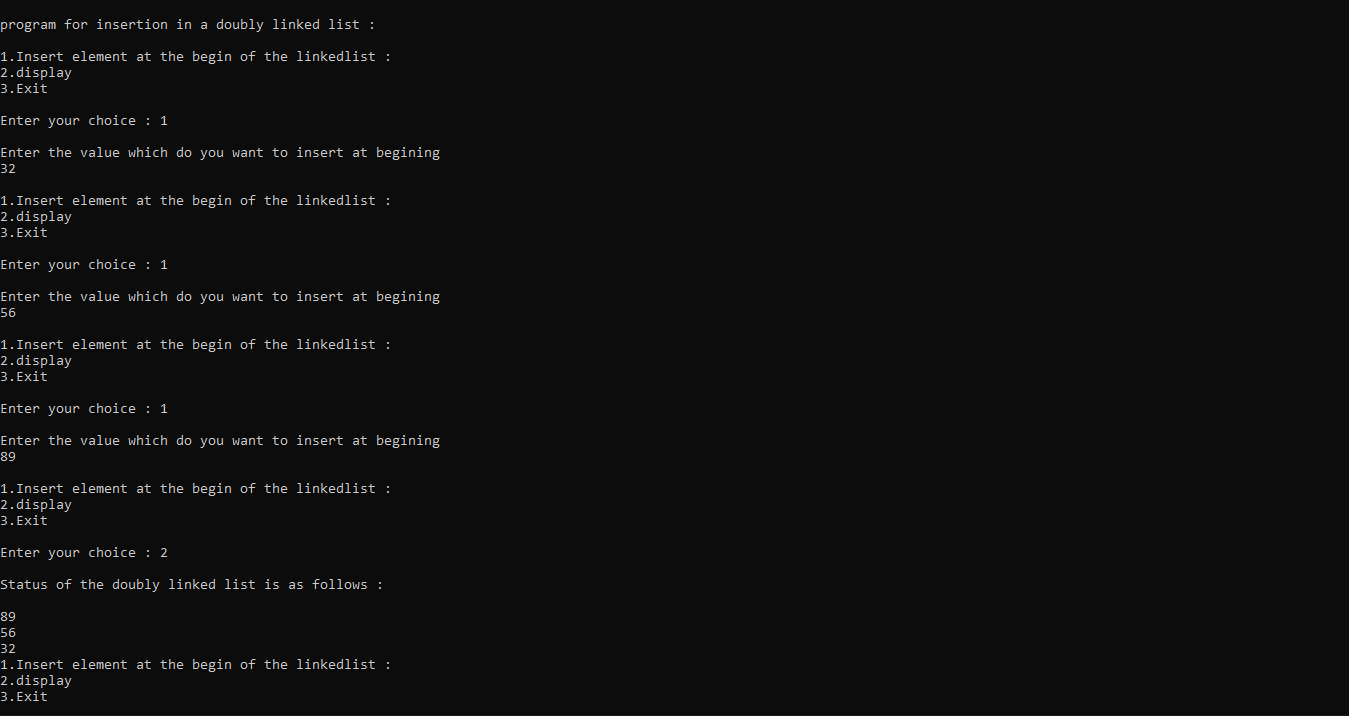
temp->next=head; head->prev=temp;

temp->prev=NULL; head=temp;

}

}

**OUTPUT:**

****

**Practical – 16**

**AIM:** Write a menu driven program to implement following operation on the doubly linked list. a) Insert a node at the end of the doubly linked list.

**PROGRAM:**

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

struct node

{ int num;

struct node \*next;

struct node \*prev;

};

struct node \*head=NULL,\*temp, \*first, \*last;

int info;

void display();

void insert\_at\_begin();

void insert\_at\_end();

int main() {

int i;

printf("\nprogram for insertion in a doubly linked list :\n");

do {

printf("\nYASH PATIL Enter your choice :\n");

printf("\n1.Insert element at the begin of the linkedlist :");

printf("\n2.Insert element at the end of the linkedlist :");

printf("\n3.display"); printf("\n4.Exit\n");

scanf("%d",&i);

switch(i) {

case 1: insert\_at\_begin();

break;

case 2:

insert\_at\_end();

break;

case 3:

display();

break;case 4: exit(0);

}

} while(1);

}

void display() {

struct node \*ptr; ptr=head;

printf("\nStatus of the doubly linked list is as follows :\n");

while(ptr!=NULL) /\* traversing the linked list \*/

{

printf("\n%d",ptr->num); ptr=ptr->next;

}

}

void insert\_at\_begin() {

printf("\nEnter the value which do you want to insert at begining\n");

scanf("%d",&info);

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

//(struct node)malloc(sizeof(NODE));

temp->num=info; temp->next=NULL;

temp->prev=NULL;

if(head==NULL) {

head=temp; last=temp;

}

else {

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

temp->prev=NULL; head=temp;

}

}

void insert\_at\_end(){

struct node \*ptr;

printf("\nEnter Elemnet to insert ");

scanf("%d",&info);

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

temp->num=info;

temp->next=NULL;

temp->prev=NULL;

if(head==NULL){

head=temp;last=temp;

}

ptr=head;

while(ptr->next!=NULL){

ptr=ptr->next;

}

ptr->next=temp;

temp->prev=ptr;

temp->next=NULL;

}

**OUTPUT:**

****

**Practical – 17**

**AIM:** Write a menu driven program to implement following operation on the doubly linked list. a) Delete last node of the doubly linked list.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

//#include<process.h>

struct node{

int num;

struct node \*next;

struct node \*prev;

};

struct node \*head=NULL,\*temp, \*first, \*last;

int info;

void insert\_at\_end();

void display();

void del\_at\_end();

int main() /\* starting the main method() \*/

{

int i;

printf("program for insertion in a doubly linked list :\n");

do {

printf("\n1.Insert element at the end of the linkedlist :");

printf("\n2.delete last node");

printf("\n3.display");

printf("\n4.Exit\n");

printf("Enter your choice : ");

scanf("%d",&i);

switch(i) {

case 1:

insert\_at\_end();

display();

break;

case 2:

del\_at\_end();

display();

break;

case 3:

display();

break;

case 4: exit(0);

}

}

while(1);

}

void display() {

struct node \*ptr;

ptr=head;

printf("\nStatus of the doubly linked list is as follows :\n");

while(ptr!=NULL) /\* traversing the linked list \*/

{

printf("\n%d",ptr->num); ptr=ptr->next;

}

}

void insert\_at\_end(){

struct node \*ptr; printf("\nEnter your element in the linked list :"); scanf("%d",&info);

temp=(struct node \*)malloc(sizeof(struct node)); /\* allocating memory for the node to be inserted \*/

temp->num=info;

temp->next=NULL;

temp->prev=NULL;

if(head==NULL) { head=temp; last=temp; }

ptr=head;

while(ptr->next!=NULL)

{ ptr=ptr->next;

}

ptr->next=temp; temp->prev=ptr; temp->next=NULL;

}

void del\_at\_end()

{

struct node \* ptr;

if(head == NULL)

{

printf(" Delete is not possible. No data in the list.\n");

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nNode Deleted\n");

}

else

{

ptr = head;

while(ptr->next != NULL)

{

ptr = ptr -> next;

}

ptr -> prev -> next = NULL;

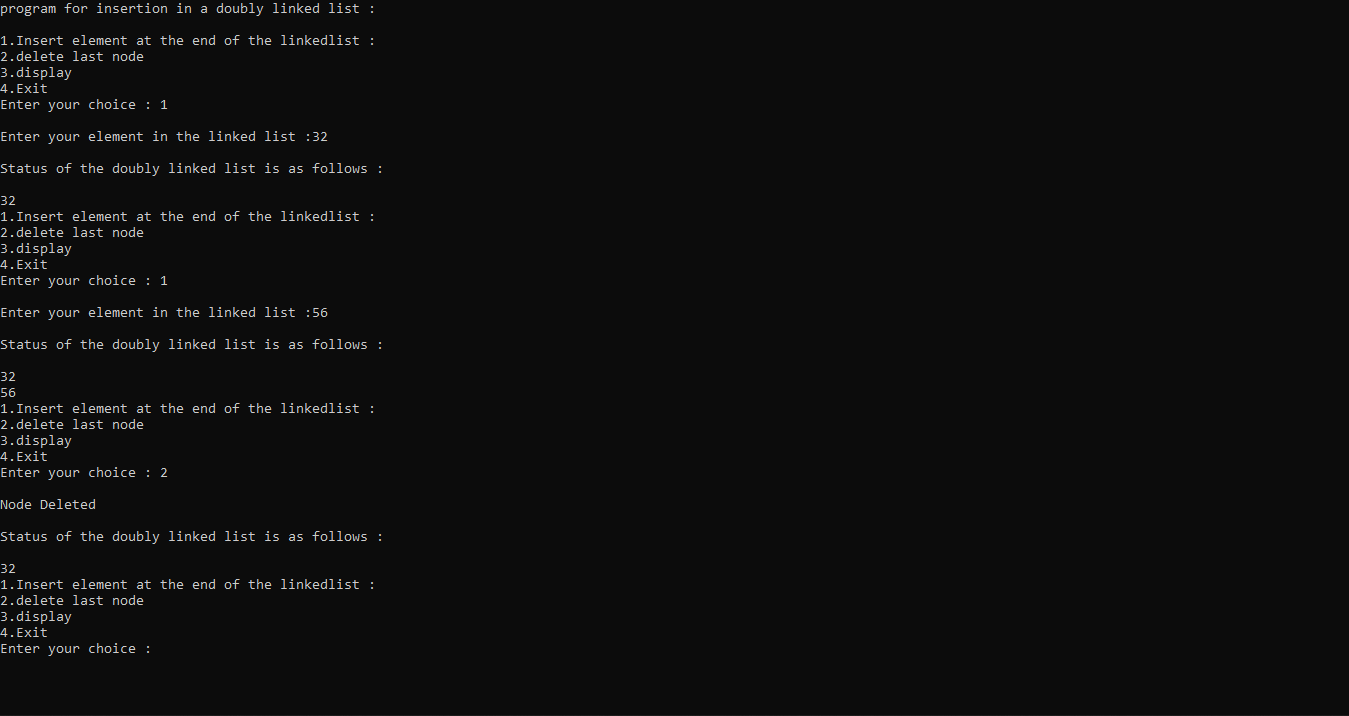
free(ptr);

printf("\nNode Deleted\n");

}

}

**OUTPUT:**

****

**Practical – 18**

**AIM:** Write a menu driven program to implement following operation on the doubly linked list. a) Delete a node after a specified position in the doubly linked list.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

//#include<process.h>

struct node{

int num;

struct node \*next;

struct node \*prev;

};

struct node \*head=NULL,\*temp, \*first, \*last;

int info;

void del\_after\_pos();

void display();

void insert\_at\_end();

int main() { /\* starting the main method() \*/

int i;

printf("\nprogram for insertion in a doubly linked list :\n");

do {

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

printf("\n1.Insert element at the end of the linkedlist :");

printf("\n2.delete node after the specified node");

printf("\n3.display");

printf("\n4.Exit\n");

scanf("%d",&i);

switch(i) {

case 1:

insert\_at\_end();

display();

break;

case 2:

del\_after\_pos();

display();

break;

case 3:

display();

break;

case 4: exit(0);

}

}

while(1);

}

void display() {

struct node \*ptr;

ptr=head;

printf("\nStatus of the doubly linked list is as follows :\n");

while(ptr!=NULL) /\* traversing the linked list \*/

{ printf("\n%d",ptr->num); ptr=ptr->next; }

}

void insert\_at\_end(){

struct node \*ptr; printf("\nEnter your element in the linked list :"); scanf("%d",&info);

temp=(struct node \*)malloc(sizeof(struct node)); /\* allocating memory for the node to be inserted \*/

temp->num=info;

temp->next=NULL;

temp->prev=NULL;

if(head==NULL) { head=temp; last=temp; }

ptr=head;

while(ptr->next!=NULL)

{ ptr=ptr->next;

}

ptr->next=temp; temp->prev=ptr; temp->next=NULL;

}

void del\_after\_pos()

{

struct node \*ptr, \*temp;

if( head == NULL)

{ printf("list is empty"); }

else{

int val;

printf("\n Enter the data after which the node is to be deleted : ");

scanf("%d", &val);

ptr = head;

while(ptr -> num != val)

ptr = ptr -> next;

if(ptr -> next == NULL)

{

printf("\nCan't delete\n");

}

else if(ptr -> next -> next == NULL)

{

ptr ->next = NULL;

}

else

{

temp = ptr -> next;

ptr -> next = temp -> next;

temp -> next -> prev = ptr;

free(temp);

printf("\nnode deleted\n");

}

}

}

**OUTPUT:**

****

**Practical – 19**

**AIM:** Write a program to implement the following operation on circular linked list

1. Insert node at end

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

//void insert\_beg();

void insert\_end();

void display();

struct node {

int data;

struct node \*next;

}\*start=NULL;

int main() {

int ch;

while(1)

{

printf("\n \*\*\*CIRCULAR LINKLIST MENU\*\*\*");

printf("\n\n1. insert\_end \n 2.Display\n 3.exit");

printf("\n\n enter your choice ");

scanf("%d",&ch);

switch(ch)

{

case 1:insert\_end();

display();

break;

case 2:display();

break;

case 3: exit(0);

break;

default:printf("\nwrong coice!");

break;

}

}

}

void insert\_end() {

int val;

struct node \*new\_node,\*ptr;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

new\_node->next=start;

}

void display()

{

struct node \*ptr;

ptr=start;

while(ptr->next!=start)

{

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

ptr=ptr->next;

}

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

}

**OUTPUT:**

****

**Practical – 20**

**AIM:** Write a program to implement the following operation on circular linked list

1. Insert node at specified position.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void insert\_beg();

void insert\_befpos();

void insert\_end();

void display();

struct node {

int data;

struct node \*next;

}\*start=NULL;

int main() {

int ch;

while(1)

{

printf("\*\*\*CIRCULAR LINKLIST MENU\*\*\*");

printf("\n\n1.insert\_end\n2. insert\_at specified pos \n 3.Display\n 4.exit");

printf("\n enter your choice ");

scanf("%d",&ch);

switch(ch)

{

//case 1:insert\_beg();

//break;

case 1:insert\_end();

break;

case 2:insert\_befpos();

break;

break;

case 3:display();

break;

case 4: exit(0);

break;

default:printf("\nwrong coice!");

break;

}

}

getch();

}

void insert\_beg() {

struct node \*new\_node,\*ptr;

int val;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

new\_node->next=start;

ptr->next=new\_node;

start=new\_node;

}

void insert\_befpos(){

struct node \*new\_node,\*ptr,\*preptr;

int val,num;

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

printf("enter the value befor which val is inserted");

scanf("%d",&num);

if(start->data == num)

{

insert\_beg();

}

else{

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

ptr=start;

while(ptr->data!=num)

{

preptr=ptr;

ptr=ptr->next;

}

new\_node->next=ptr;

preptr->next=new\_node;

}

}

void insert\_end() {

int val;

struct node \*new\_node,\*ptr;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

new\_node->next=start;

}

void display()

{

struct node \*ptr;

ptr=start;

while(ptr->next!=start)

{

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

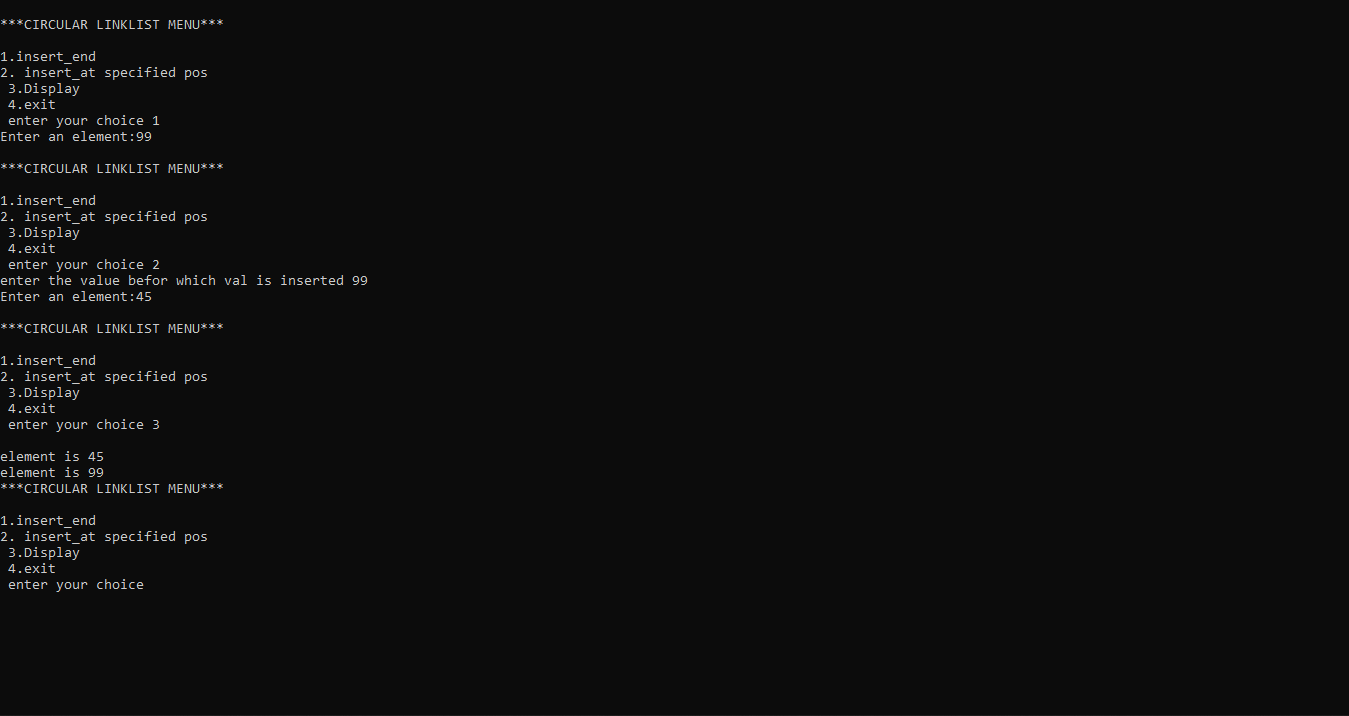
ptr=ptr->next;

}

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

}

**OUTPUT:**

****

**Practical – 21**

**AIM:** Write a program to implement the following operation on circular linked list a) Delete the first node

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void delete\_first();

void insert\_end();

void display();

struct node {

int data;

struct node \*next;

}\*start=NULL;

int main() {

int ch;

while(1)

{

printf("\n \*\*\*CIRCULAR LINKLIST MENU\*\*\*");

printf("\n\n1.insert\_end\n2. delete first \n 3.Display\n 4.exit");

printf("\n\n enter your choice ");

scanf("%d",&ch);

switch(ch)

{

case 1:insert\_end();

break;

case 2:delete\_first();

break;

case 3:display();

break;

case 4: exit(0);

break;

default:printf("\nwrong coice!");

break;

}

}

getch();

}

void insert\_end() {

int val;

struct node \*new\_node,\*ptr;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

new\_node->next=start;

}

void display()

{

struct node \*ptr;

ptr=start;

while(ptr->next!=start)

{

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

ptr=ptr->next;

}

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

}

void delete\_first()

{

struct node \*prev=start,\*first=start;

if(start == NULL)

{

printf("list empty");

}

else if(prev->next == prev)

{

start=NULL;

}

else{

while(prev->next != start)

{

prev=prev->next;

}

prev->next = first->next;

start=prev->next;

free(first);

}

}

**OUTPUT:**

****

**Practical – 22**

**AIM:** Write a program to implement the following operation on circular linked list a) Delete the last node

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void delete\_last();

void insert\_end();

void delete\_first();

void display();

struct node {

int data;

struct node \*next;

}\*start=NULL;

int main(){

int ch;

while(1){

printf("\n \*\*\*CIRCULAR LINKLIST MENU\*\*\*");

printf("\n\n1.insert\_end\n2. delete last\n 3.delete first \n 4.Display\n 5.exit");

printf("\n\n Enter your choice ");

scanf("%d",&ch);

switch(ch)

{

case 1:insert\_end();

break;

case 2:delete\_last();

break;

case 3:delete\_first();

break;

case 4:display();

break;

case 5: exit(0);

break;

default:printf("\nwrong coice!");

break;

}

}

}

void insert\_end() {

int val;

struct node \*new\_node,\*ptr;

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

printf("Enter an element:");

scanf("%d",&val);

new\_node->data=val;

if(start==NULL) //If list is empty

{

start=new\_node;

}

else

{

ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

ptr->next=new\_node;

}

new\_node->next=start;

display();

}

void display(){

struct node \*ptr;

ptr=start;

while(ptr->next!=start)

{

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

ptr=ptr->next;

}

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

}

void delete\_last(){

struct node \*ptr, \*preptr;

if(start==NULL)

{

printf("\nUNDERFLOW\n");

}

else if (start ->next == start)

{

start= NULL;

free(start);

printf("\nNode Deleted\n");

}

else

{

ptr = start;

while(ptr ->next != start)

{

preptr=ptr;

ptr = ptr->next;

}

preptr->next = ptr -> next;

free(ptr);

printf("\nNode Deleted\n");

display();

}

}

void delete\_first(){

struct node \*prev=start,\*first=start;

if(start == NULL)

{

printf("list empty");

}

else if(prev->next == prev)

{

start=NULL;

}

else{

while(prev->next != start)

{

prev=prev->next;

}

prev->next = first->next;

start=prev->next;

free(first);

display();

}

}

**OUTPUT:**

****

**Practical – 23**

**AIM:** Implement recursive or non-recursive tree traversing methods of inorder

traversal.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct btnode

{

int value;

struct btnode \*l;

struct btnode \*r;

}\*root = NULL, \*temp = NULL, \*t2, \*t1;

void insert();

void inorder(struct btnode \*t);

void create();

void search(struct btnode \*t);

int main()

{

int ch;

printf("\nOPERATIONS ---");

printf("\n1 - Insert an element into tree\n");

printf("2 - Inorder Traversal\n");

printf("3 - Exit\n");

while(1)

{

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

scanf("%d", &ch);

switch (ch)

{

case 1:

insert();

break;

case 2:

inorder(root);

break;

case 3:

exit(0);

default :

printf("Wrong choice, Please enter correct choice ");

break;

}

}

}

/\* To insert a node in the tree \*/

void insert()

{

create();

if (root == NULL)

root = temp;

else

search(root);

}

/\* To create a node \*/

void create()

{

int data;

printf("Enter data of node to be inserted : ");

scanf("%d", &data);

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

temp->value = data;

temp->l = temp->r = NULL;

}

/\* Function to search the appropriate position to insert the new node \*/

void search(struct btnode \*t)

{

if ((temp->value > t->value) && (t->r != NULL)) /\* value more than root node value insert at right \*/

search(t->r);

else if ((temp->value > t->value) && (t->r == NULL))

t->r = temp;

else if ((temp->value < t->value) && (t->l != NULL)) /\* value less than root node value insert at left \*/

search(t->l);

else if ((temp->value < t->value) && (t->l == NULL))

t->l = temp;

}

/\* recursive function to perform inorder traversal of tree \*/

void inorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

if (t->l != NULL)

inorder(t->l);

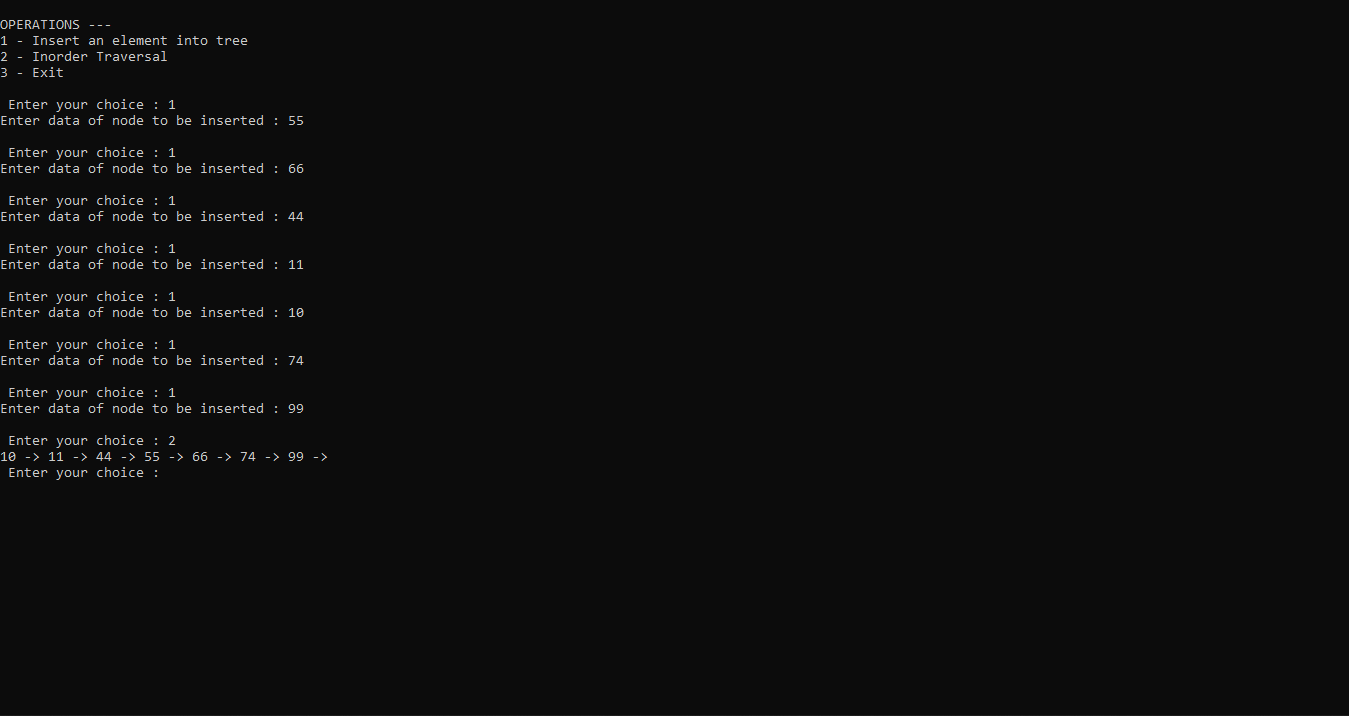
printf("%d -> ", t->value);

if (t->r != NULL)

inorder(t->r);

}

**OUTPUT:**

****

**Practical – 24**

**AIM:** Implement recursive or non-recursive tree traversing methods of Preorder

traversal.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct btnode

{

int value;

struct btnode \*l;

struct btnode \*r;

}\*root = NULL, \*temp = NULL, \*t2, \*t1;

void insert();

void inorder(struct btnode \*t);

void preorder(struct btnode \*t);

void create();

void search(struct btnode \*t);

int main()

{

int ch;

printf("\nOPERATIONS ---");

printf("\n1 - Insert an element into tree\n");

printf("2 - Inorder Traversal\n");

printf("3 - Preorder Traversal\n");

printf("4 - Exit\n");

while(1)

{

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

scanf("%d", &ch);

switch (ch)

{

case 1:

insert();

break;

case 2:

inorder(root);

break;

case 3:

preorder(root);

break;

case 4:

exit(0);

default :

printf("Wrong choice, Please enter correct choice ");

break;

}

}

}

/\* To insert a node in the tree \*/

void insert()

{

create();

if (root == NULL)

root = temp;

else

search(root);

}

/\* To create a node \*/

void create()

{

int data;

printf("Enter data of node to be inserted : ");

scanf("%d", &data);

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

temp->value = data;

temp->l = temp->r = NULL;

}

/\* Function to search the appropriate position to insert the new node \*/

void search(struct btnode \*t)

{

if ((temp->value > t->value) && (t->r != NULL)) /\* value more than root node value insert at right \*/

search(t->r);

else if ((temp->value > t->value) && (t->r == NULL))

t->r = temp;

else if ((temp->value < t->value) && (t->l != NULL)) /\* value less than root node value insert at left \*/

search(t->l);

else if ((temp->value < t->value) && (t->l == NULL))

t->l = temp;

}

/\* recursive function to perform inorder traversal of tree \*/

void inorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

if (t->l != NULL)

inorder(t->l);

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

if (t->r != NULL)

inorder(t->r);

}

void preorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

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

if (t->l != NULL)

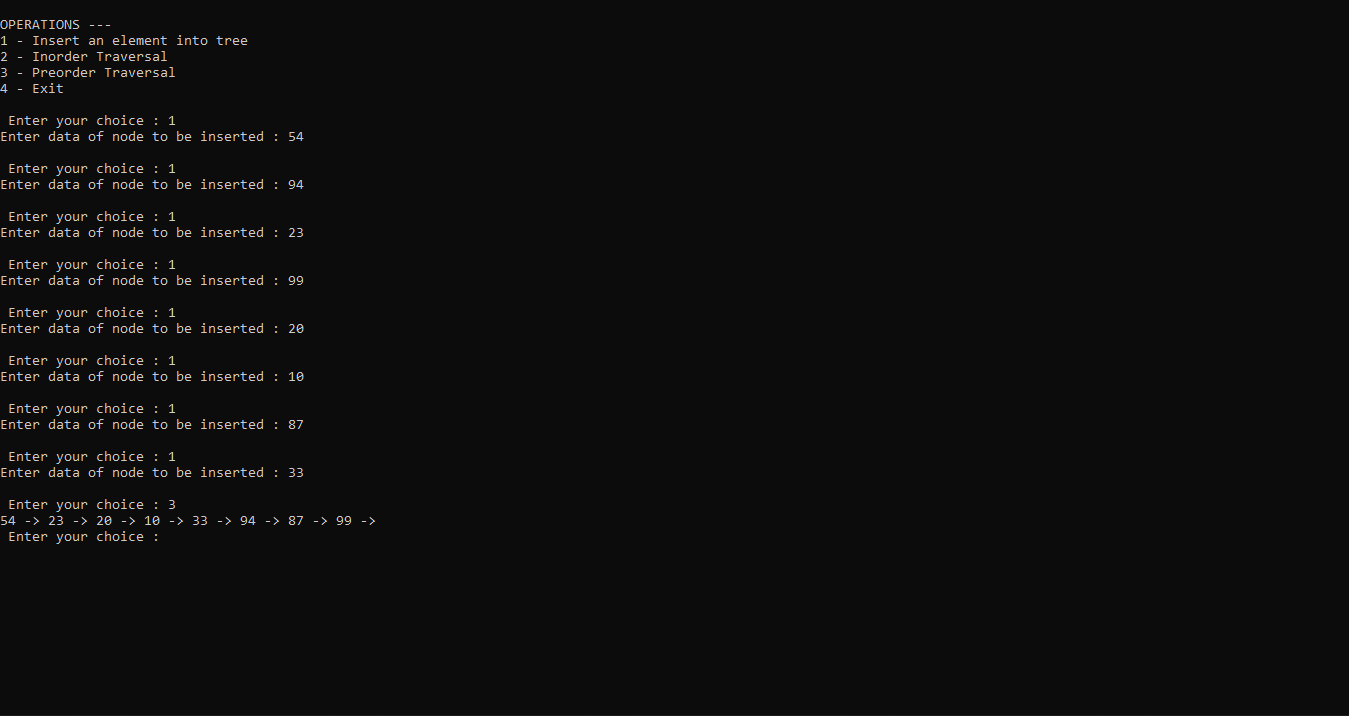
preorder(t->l);

if (t->r != NULL)

preorder(t->r);

}

**OUTPUT:**

****

**Practical – 25**

**AIM:** Implement recursive or non-recursive tree traversing methods of postorder

traversal.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct btnode

{

int value;

struct btnode \*l;

struct btnode \*r;

}\*root = NULL, \*temp = NULL, \*t2, \*t1;

void insert();

void inorder(struct btnode \*t);

void preorder(struct btnode \*t);

void postorder(struct btnode \*t);

void create();

void search(struct btnode \*t);

int main()

{

int ch;

printf("\nOPERATIONS ---");

printf("\n1 - Insert an element into tree\n");

printf("2 - Inorder Traversal\n");

printf("3 - Preorder Traversal\n");

printf("4 - Postorder Traversal\n");

printf("5 - Exit\n");

while(1)

{

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

scanf("%d", &ch);

switch (ch)

{

case 1:

insert();

break;

case 2:

printf("\nInorder Traversal\n");

inorder(root);

break;

case 3:

printf("\nPreorder Traversal\n");

preorder(root);

break;

case 4:

printf("\nPostorder Traversal\n");

postorder(root);

break;

case 5:

exit(0);

default :

printf("Wrong choice, Please enter correct choice ");

break;

}

}

}

/\* To insert a node in the tree \*/

void insert()

{

create();

if (root == NULL)

root = temp;

else

search(root);

}

/\* To create a node \*/

void create()

{

int data;

printf("Enter data of node to be inserted : ");

scanf("%d", &data);

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

temp->value = data;

temp->l = temp->r = NULL;

}

/\* Function to search the appropriate position to insert the new node \*/

void search(struct btnode \*t)

{

if ((temp->value > t->value) && (t->r != NULL)) /\* value more than root node value insert at right \*/

search(t->r);

else if ((temp->value > t->value) && (t->r == NULL))

t->r = temp;

else if ((temp->value < t->value) && (t->l != NULL)) /\* value less than root node value insert at left \*/

search(t->l);

else if ((temp->value < t->value) && (t->l == NULL))

t->l = temp;

}

/\* recursive function to perform inorder traversal of tree \*/

void inorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

if (t->l != NULL)

inorder(t->l);

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

if (t->r != NULL)

inorder(t->r);

}

void preorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

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

if (t->l != NULL)

preorder(t->l);

if (t->r != NULL)

preorder(t->r);

}

void postorder(struct btnode \*t){

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

if (t->l != NULL)

postorder(t->l);

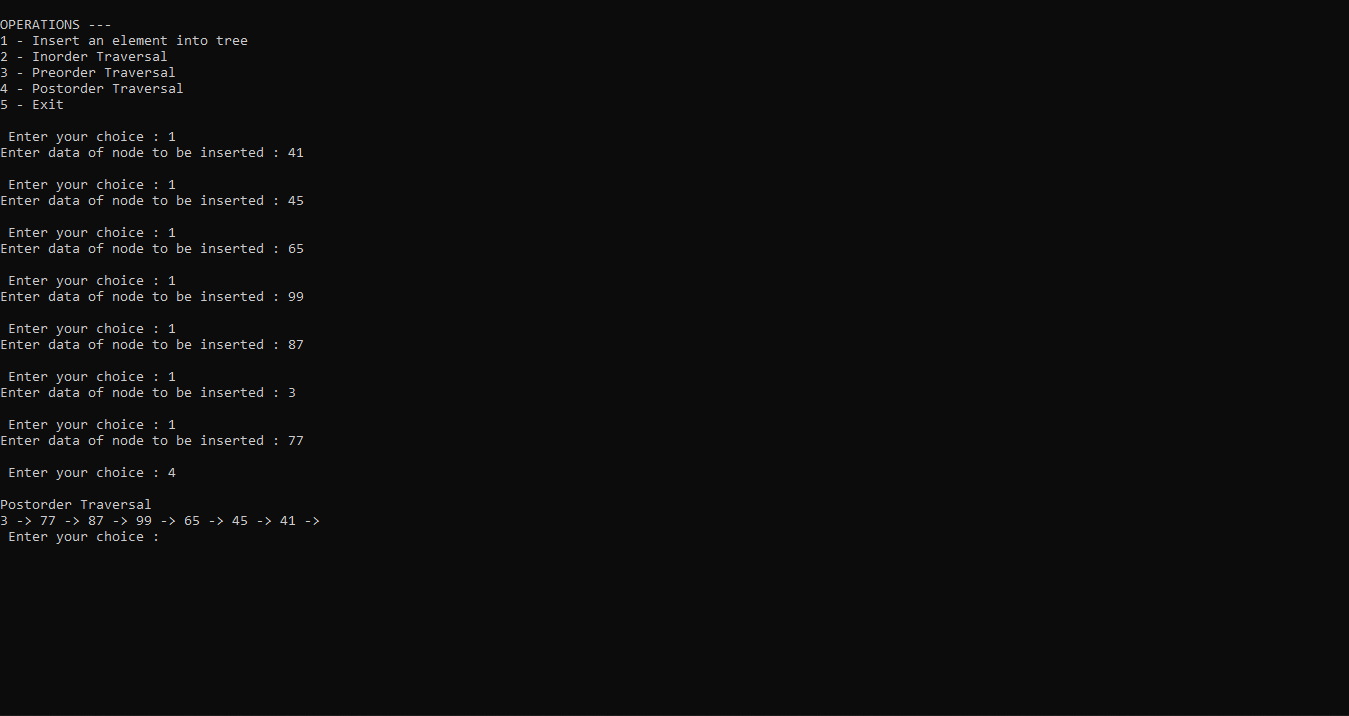
if (t->r != NULL)

postorder(t->r);

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

}

**OUTPUT:**

****

**Practical – 26**

**AIM:** Write a program to implement Merge Sort

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#define MAX 50

void mergeSort(int arr[],int low,int mid,int high);

void partition(int arr[],int low,int high);

int main(){

int merge[MAX],i,n;

printf("YASH PATIL 19CE032\n");

printf("Enter the total number of elements: ");

scanf("%d",&n);

printf("Enter the elements which to be sort: \n");

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

scanf("%d",&merge[i]);

}

partition(merge,0,n-1);

printf("After merge sorting elements are: ");

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

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

}

return 0;

}

void partition(int arr[],int low,int high){

int mid;

if(low<high){

mid=(low+high)/2;

partition(arr,low,mid);

partition(arr,mid+1,high);

mergeSort(arr,low,mid,high);

}

}

void mergeSort(int arr[],int low,int mid,int high){

int i,m,k,l,temp[MAX];

l=low;

i=low;

m=mid+1;

while((l<=mid)&&(m<=high)){

if(arr[l]<=arr[m]){

temp[i]=arr[l];

l++;

}

else{

temp[i]=arr[m];

m++;

}

i++;

}

if(l>mid){

for(k=m;k<=high;k++){

temp[i]=arr[k];

i++;

}

}

else{

for(k=l;k<=mid;k++)

{

temp[i]=arr[k];

i++;

}

}

for(k=low;k<=high;k++)

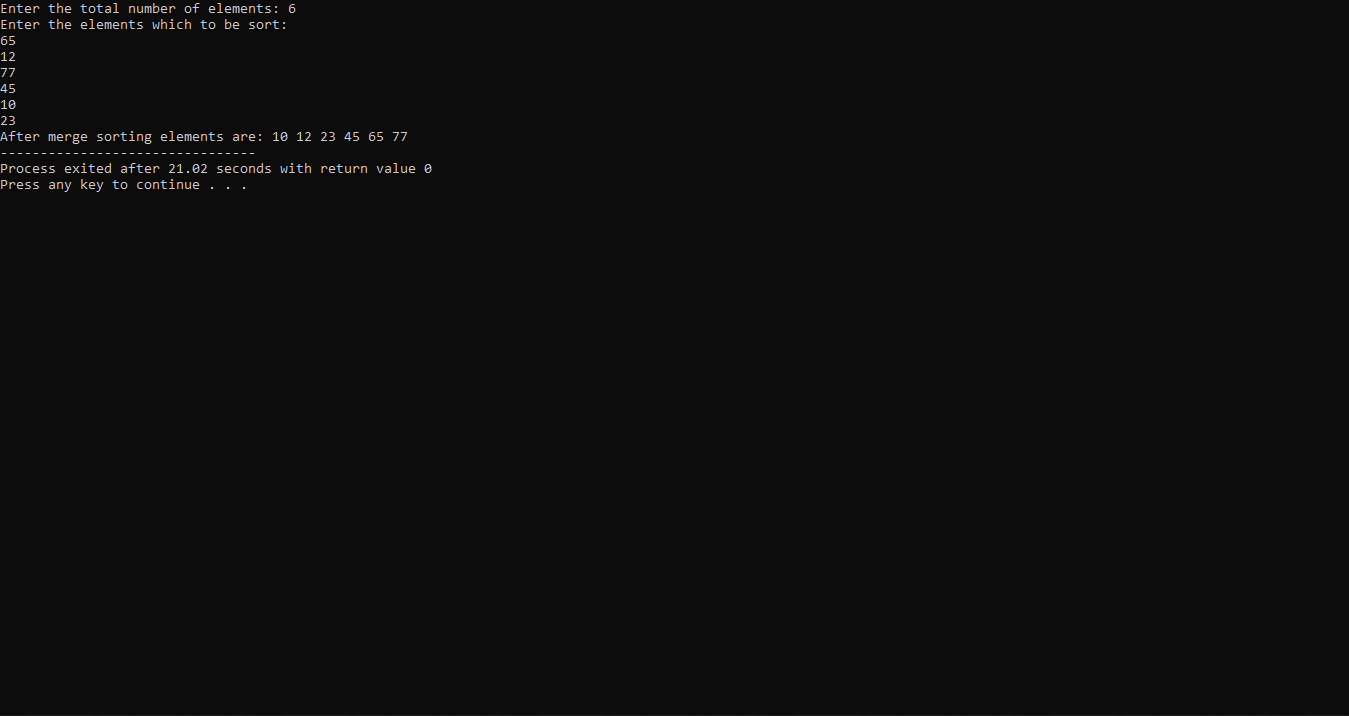
{

arr[k]=temp[k];

}

}

**OUTPUT:**

****

**Practical – 27**

**AIM:** Write a program to implement Bubble Sort

**PROGRAM:**

#include <stdio.h>

int main()

{

int i, n, temp, j, arr[10];

printf("\n Enter the number of elements in the array : ");

scanf("%d", &n);

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

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

{

scanf("%d", &arr [i]);

}

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

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

{

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

{

temp = arr[j];

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

arr[j+1] = temp;

}

}

}

printf("\n The array sorted in ascending order is :\n");

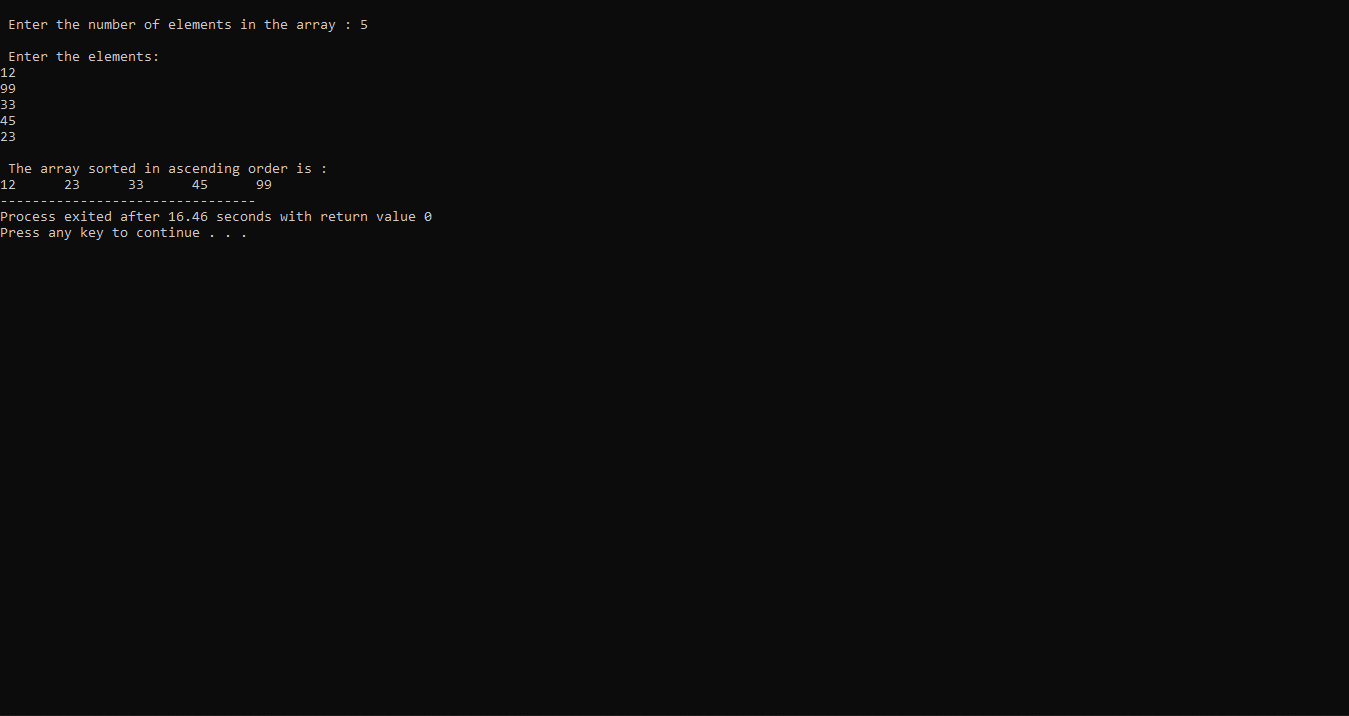
for(i=0;i<n;i++)

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

//return 0;

}

**OUTPUT:**

****

**Practical – 28**

**AIM:** Write a program to implement Selection Sort

**PROGRAM:**

#include<stdio.h>

int main(){

int i, j, count, temp, number[25];

printf("Enter number of elements: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

for(i=0;i<count;i++)

scanf("%d",&number[i]);

for(i=0;i<count;i++){

for(j=i+1;j<count;j++){

if(number[i]>number[j]){

temp=number[i];

number[i]=number[j];

number[j]=temp;

}

}

}

printf("Sorted elements: ");

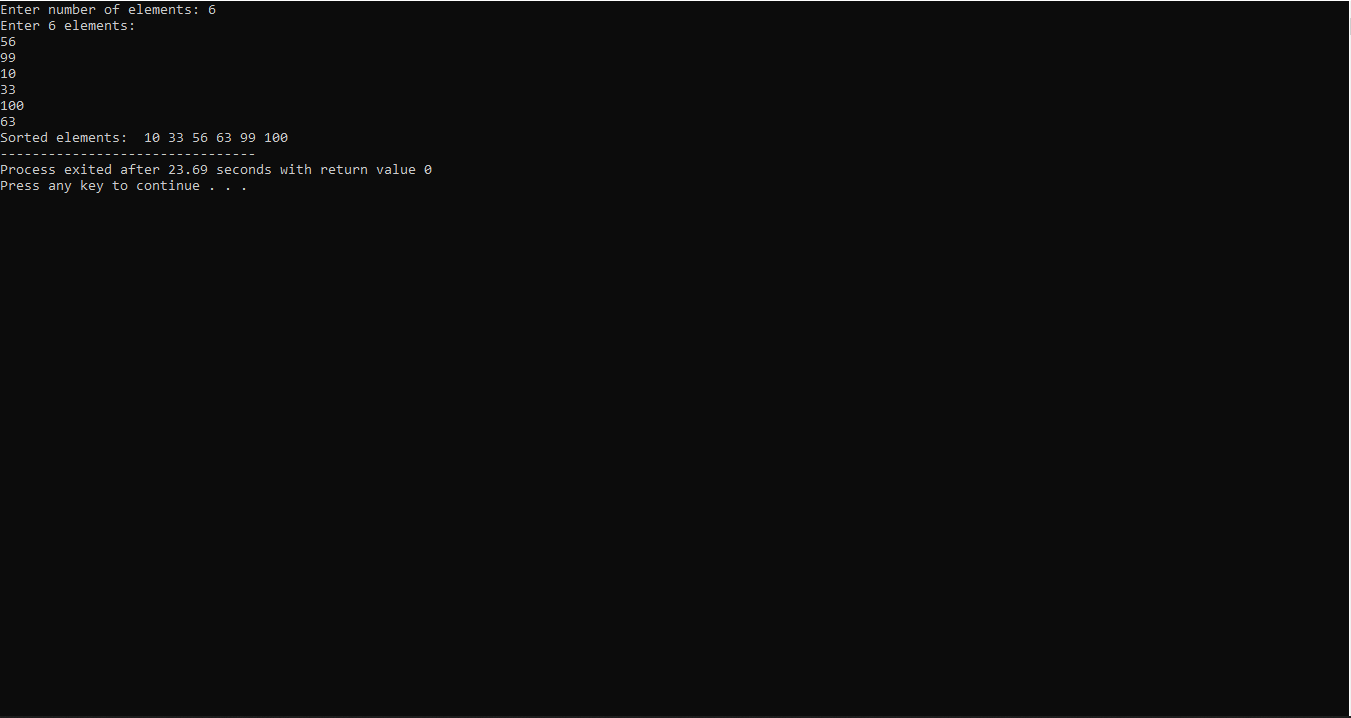
for(i=0;i<count;i++)

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

return 0;

}

**OUTPUT:**

****