**Experiment No. 1:**

**Write a C program to perform matrix multiplication using array.**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int a[10][10],b[10][10],c[10][10],i,j,k,m,n,p,q;

clrscr();  
printf("Enter The Rows And Cloumns of The 1st Matrix:");   
scanf("%d %d",&m,&n);   
printf("\nEnter The Rows And Cloumns of The 2nd Matrix:");   
scanf("%d %d",&p,&q);   
printf("\nEnter Elements of The 1st Matrix:\n");   
for(i=0;i< m;i++)   
{

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

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

}   
printf("\nEnter Elements of The 2nd Matrix:\n");   
for(i=0;i< p;i++)

{

for(j=0;j< q;j++)

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

}   
printf("1st Matrix Is:\n");  
for(i=0;i< m;i++)

{

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

printf(" %d ",a[i][j]);

printf("\n");   
}   
printf("2nd Matrix Is:\n");

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

{

for(j=0;j< q;j++)

printf(" %d ",b[i][j]);

printf("\n");   
}   
if(n!=p)

{

printf("Aborting./nMultiplication Of The Above Matrices Not Possible.");

exit(0);   
}   
else

{

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

{

for(j=0;j< q;j++)

{

c[i][j] = 0;

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

c[i][j] += a[i][k] \* b[k][j];

}

}

printf("\nThe Product Of The Two Matrices Is:\n\n");

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

{

for(j=0;j< q;j++)

printf(" %d ",c[i][j]);

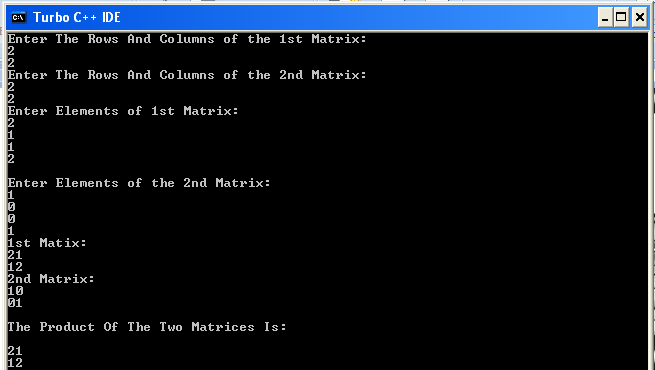
printf("\n");

}   
}

getch();

}

Output:



**Experiment No. 2:**

**a) Write a C program to create a stack using an array and perform**

**(i) push operation (ii) pop operation (iii) display operation**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int size,\*st,choice,top=-1,t,i;

int overflow(int,int);

int underflow(int);

clrscr();

printf("Enter size.\n");

scanf("%d",&size);

st=(int \*)calloc(size,2);

do

{

printf("\n Choose from menu:\n 1. Push\n 2. Pop\n 3. Display\n 4. Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(overflow(top,size))

printf("\nOverflow\n");

else

{

top++;

printf(“Enter data.\n”);

scanf("%d",(st+top));

}

break;

case 2:

if(underflow(top))

printf("\nUnderflow\n");

else

{

t=\*(st+top);

top--;

printf("%d is popped\n",t);

}

break;

case 3:

if(underflow(top))

printf("\nEmpty Stack\n");

else

{

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

printf("\n %d",\*(st+top));

}

break;

}

}while(choice<4);

getch();

}

int overflow(int T,int S)

{

return(T==(S-1)?1:0);

}

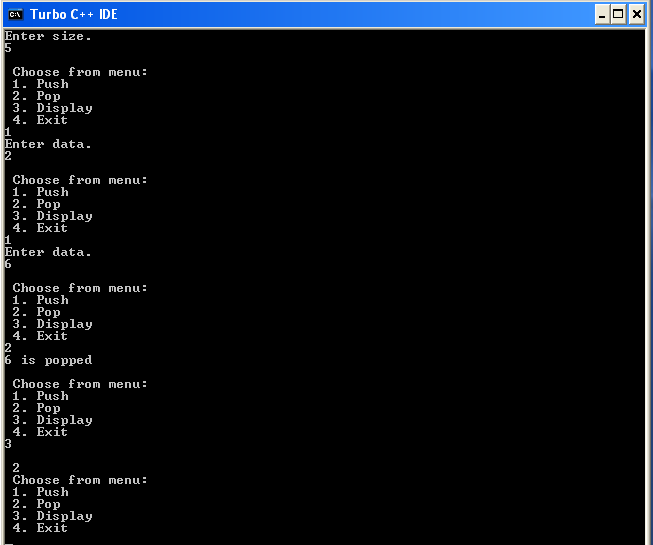
int underflow(int T)

{

return(T==-1?1:0);

}

Output:



**b) Write a C program to create a queue and perform**

**i) Enqueue ii) Dequeue iii) Traversal**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int size,\*st,choice,rear=-1,front=-1,t,i;

int overflow(int,int);

int underflow(int);

clrscr();

printf("Enter size.\n");

scanf("%d",&size);

st=(int \*)calloc(size,2);

do

{

printf("Choose: 1. Enqueue\n 2. Dequeue\n 3. Display\n 4. Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(overflow(rear,size))

printf("\nOverflow.\n");

else

{

rear++;

printf(“Enter data.\n”);scanf("%d",(st+rear));

if(front==-1)

front=0;

}

break;

case 2:

if(underflow(front))

printf("\nUnderflow.\n");

else

{

t=\*(st+front);

front++;

printf("%d is dequeued.\n",t);

if(front>rear)

{

front=-1;

rear=-1;

}

}

break;

case 3:

if(underflow(front))

printf("\nUnderflow.\n");

else

{

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

printf("\n %d",\*(st+i));

}

break;

}

}while(choice<4);

}

int overflow(int R,int S)

{

return(R==(S-1)?1:0);

}

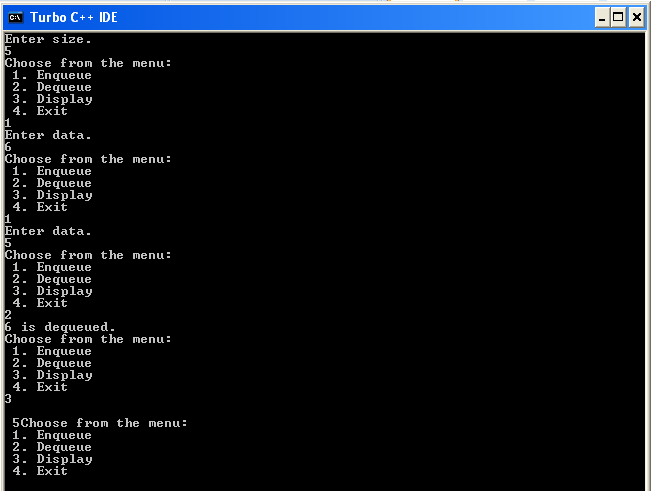
int underflow(int F)

{

return(F==-1?1:0);

}

Output:



**c) Write a C program to create a priority queue and perform:**

**i) Enqueue operation ii) Dequeue operation iii) Display operation**

Coding:

#include<stdio.h>

struct que

{

int data;

int prio;

int order;

};

void main()

{

int front=-1,rear=-1,pos,choice,size=10,i;

struct que \*q,temp;

int overflow(int,int);

int underflow(int);

int find(struct que q[],int,int,int);

do

{

printf("Choose:\n 1. Enqueue\n 2. Dequeue\n 3. Display\n 4. Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

if(overflow(rear,size))

printf("Overflow");

else

{

printf(“Enter Data.\n”);

scanf(“%d”,&temp.data);

printf("Enter Priority.\n");

scanf("%d",&temp.prio);

printf("Enter Order.\n");

scanf("%d",&temp.order);

if(rear==-1)

{

rear++;

front=0;

q[rear]=temp;

}

else

{

pos=find(q,rear,temp.prio,temp.order);

if(pos>rear)

{

rear++;

q[rear]=temp;

}

else

{

rear++;

for(i=rear;i>pos;i--)

q[i]=q[i-1];

q[pos]=temp;

}

}

}break;

case 2:

if(underflow(front))

printf("\nUnderflow.\n");

else

{

temp=q[front];

front++;

printf("\n%d is dequeued.\n",temp);

if(front>rear)

{

front=-1;

rear=-1;

}

}break;

case 3:

if(underflow(front))

printf("\nEmpty Queue.\n");

else

{

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

{

printf("\n %d",q[i].data);

pritnf("\n %d",q[i].prio);

printf("\n %d",q[i].order);

}

}break;

}

}while(choice<4);

}

int overflow(int R,int S)

{

return(R==(S-1)?1:0);

}

int underflow(int F)

{

return(F==-1?1:0);

}

int find(struct que q1[10],int R,int P,int O)

{

int i;

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

{

if(q1[i].prio<P)

return i;

else if(q1[i].prio==P && q1[i].order>0)

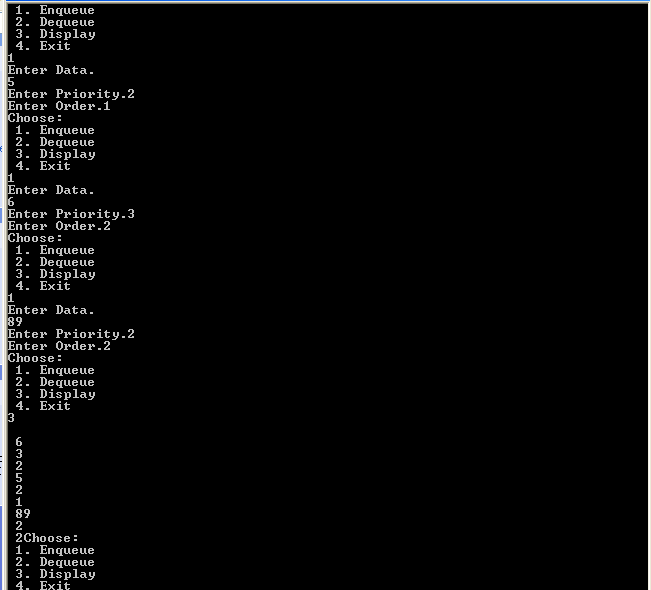
return i;

}

return i;

}

Output:



**Experiment 3:**

**Write a C program that uses Stack operations to perform the following:**

**i) Converting infix expression into postfix operation**

**ii) Evaluating the postfix expression**

**i)** Coding:

# include <stdio.h>

# include <conio.h>

# include <ctype.h>

# include<string.h>

char expr[50], oexpr[50], opstack[20];

int top=-1;

void push(char ch)

{

top++;

opstack[top] = ch;

}

char pop()

{

char c = opstack[top];

top--;

return c;

}

int priority(char ch)

{

if(ch == '(')

return 0;

else if (ch == '+' || ch == '-')

return 1;

else if(ch == '\*' || ch == '/')

return 2;

else if(ch == '^')

return 3;

else if (ch == ')')

return 4;

}

void convert(char s[])

{

char c, opr;

int i, pos = 0, l = strlen(s), cur\_prio, prev\_prio;

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

{

c = s[i];

if(isalpha(c) || isdigit(c))

oexpr[pos++] = c;

else if(c == '(')

push(c);

else

{

if(top == -1)

push(c);

else if(c != ')')

{

opr = pop();

cur\_prio = priority(c);

prev\_prio = priority(opr);

if(cur\_prio > prev\_prio)

{

push(opr);

push(c);

}

else

{

while(cur\_prio <= prev\_prio)

{

oexpr[pos++] = opr;

opr = pop();

prev\_prio = priority(opr);

}

push(opr);

push(c);

}

}

else if(c == ')')

{

opr = pop();

while(opr != '(')

{

oexpr[pos++] = opr;

opr = pop();

}

}

}

}

while(top > -1)

oexpr[pos++] = pop();

printf("\n The expression is :\n %s",oexpr);

}

void main()

{

clrscr();

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

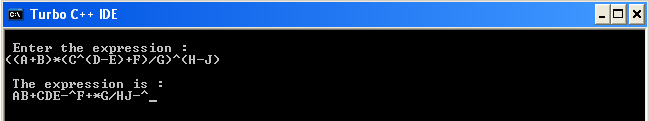
gets(expr);

convert(expr);

getch();

}

Output:



**ii)** Coding:

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<ctype.h>

int top=-1,st[10];

int convert(char ch)

{

int n1;

char n[2];

n[0]=ch;

n[1]='\0';

n1=atoi(n);

return n1;

}

void push(int n)

{

st[++top]=n;

}

int pop()

{

return(st[top--]);

}

void main()

{

char inp[50];

int i,j,res,n1,n2,n;

clrscr();

printf("Enter data.\n");

scanf("%s",inp);

n=strlen(inp);

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

{

if(isdigit(inp[i]))

{

j=convert(inp[i]);

push(j);

}

else

{

n1=pop();

n2=pop();

if(inp[i]=='+')

res=n1+n2;

else if(inp[i]=='-')

res=n1-n2;

else if(inp[i]=='\*')

res=n1\*n2;

else if(inp[i]=='/')

res=n1/n2;

push(res);

}

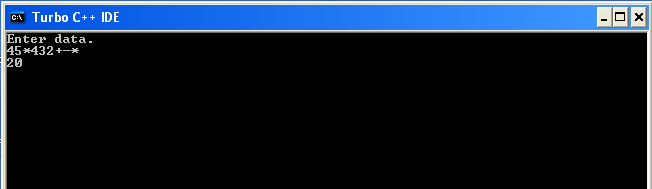
}

printf("%d",res);

getch();

}

Output:



**Experiment 4:**

**Write a C program that uses functions to perform the following operations on Single linked list:**

**i) Creation ii) Insertion iii) Deletion iv) Traversal in both ways**

Coding:

#include<stdio.h>

#include<conio.h>

struct student

{

int rollno;

char name[40];

int mark;

struct student \*next;

};

struct student \*ins\_at\_beg(struct student \*s)

{

struct student \*temp;

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\n enter student details:\n");

printf("\n enter roll no:");

scanf("%d",&temp->rollno);

printf("\n enter name:");

fflush(stdin);

gets(temp->name);

printf("\n enter marks:");

scanf("%d",&temp->mark);

temp->next=s;

return temp;

}

void ins\_at\_end(struct student \*s)

{

struct student \*temp,\*cur;

if(s==NULL)

printf("\n list is empty. Try option 1.\n");

else

{

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\n enter student details:\n");

printf("\n enter roll no:");

scanf("%d",&temp->rollno);

printf("\n enter name:");

fflush(stdin);

gets(temp->name);

printf("\n enter marks:");

scanf("%d",&temp->mark);

cur=s;

while(cur->next!=NULL)

cur=cur->next;

cur->next=temp;

}

}

void display(struct student \*s)

{

struct student \*temp,\*cur;

if(s==NULL)

printf("\n list is empty");

else

{

while(cur!=NULL)

{

printf("\nstudent details:\n");

printf("\nroll no:%d",cur->rollno);

printf("\nname:%s",cur->name);

printf("\nmarks:%d",cur->mark);

cur=cur->next;

}

}

}

struct student \*del\_at\_beg(struct student \*s)

{

struct student \*temp;

if(s==NULL)

printf("\nlist is empty");

else

{

temp=s;

s=temp->next;

free(temp);

printf("\n first node deleted");

}

return s;

}

void del\_at\_end(struct student \*s)

{

struct student \*prev,\*cur;

if(s==NULL)

printf("\n list is empty");

else

{

cur=s;

while(cur->next!=NULL)

{

prev=cur;

cur=cur->next;

}

prev->next=NULL;

free(cur);

}

}

void ins\_in\_btw(struct student \*s)

{

int rn;

struct student \*temp,\*cur;

if(s==NULL)

printf("\nlist is empty. Try option 1.\n");

else

{

printf("enter roll number to be searched for:");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\nenter roll no.:");

scanf("%d",&temp->rollno);

printf("\nenter name:");

fflush(stdin);

gets(temp->name);

temp->next=cur->next;

cur->next=temp;

break;

}

else

cur=cur->next;

}

}

if(cur==NULL)

printf("\n roll no. not found");

}

void del\_in\_btw(struct student \*s)

{

int rn;

struct student \*temp,\*cur;

if(s==NULL)

printf("\nlist is empty");

else

{

printf("enter roll number to be searched for:");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

temp->next=cur->next;

free(cur);

break;

}

else

{

temp = cur;

cur=cur->next;

}

}

}

if(cur==NULL)

printf("\n roll no. not found");

}

void search(struct student \*s)

{

int rn;

struct student \*cur;

if(s==NULL)

printf("\n list is empty");

else

{

printf("\n enter roll no. to be searched for");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

printf("\nstudent details:\n");

printf("\nroll no:%d",cur->rollno);

printf("\nname:%s",cur->name);

printf("\nmarks:%d",cur->mark);

}

cur=cur->next;

}

if(cur==NULL)

printf("\n roll no. not found");

}

}

void main()

{

int ch;

struct student \*start=NULL;

clrscr();

do

{

printf("\nChoose:");

printf("\n1. Insertion in beginning ");

printf("\n2. Insertion at end");

printf("\n3. Insertion in between");

printf("\n4. Display");

printf("\n5. Deletion in beginning");

printf("\n6. Deletion in end");

printf("\n7. Deletion in between");

printf("\n8. Search");

printf("\n9. Exit");

scanf("%d",&ch);

switch(ch)

{

case 1:

start=ins\_at\_beg(start);

break;

case 2:

ins\_at\_end(start);

break;

case 3:

ins\_in\_btw(start);

break;

case 4:

display(start);

break;

case 5:

start=del\_at\_beg(start);

break;

case 6:

del\_at\_end(start);

break;

case 7:

del\_in\_btw(start);

break;

case 8:

search(start);

break;

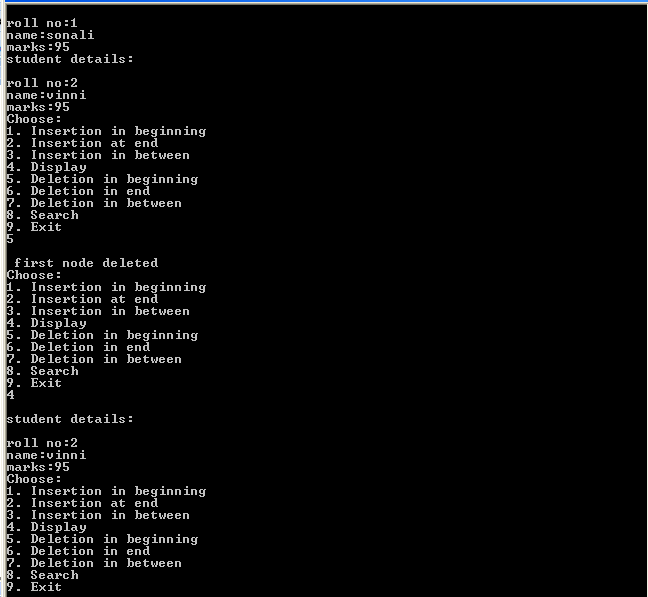
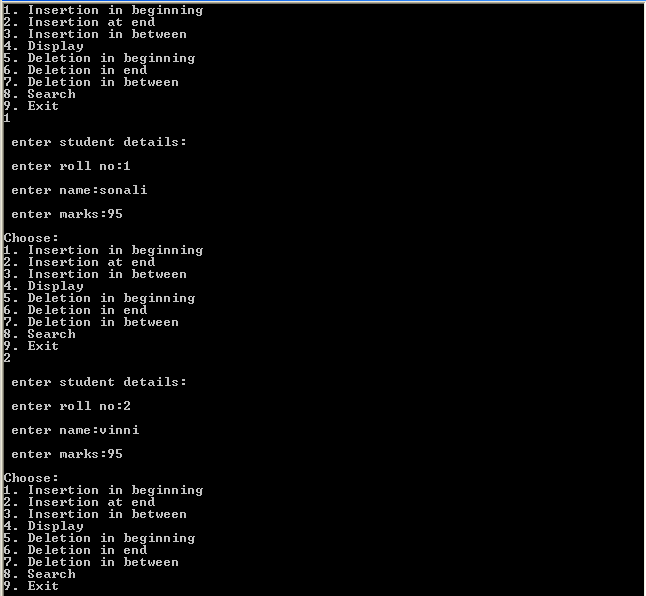
}

}while(ch<8);

getch();

}

Output:



**Experiment 5:**

**Write a C program that uses functions to perform the following operations on Double linked list:**

**i) Creation ii) Insertion iii) Deletion iv) Traversal**

Coding:

#include<stdio.h>

#include<conio.h>

struct student

{

int rollno;

char name[40];

int mark;

struct student \*next;

struct student \*prev;

};

struct student \*ins\_at\_beg(struct student \*s)

{

struct student \*temp;

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\n enter student details:\n");

printf("\n enter roll no:");

scanf("%d",&temp->rollno);

printf("\n enter name:");

fflush(stdin);

gets(temp->name);

printf("\n enter marks:");

scanf("%d",&temp->mark);

temp->next=s;

if(s!=NULL)

s->prev=temp;

return temp;

}

void ins\_at\_end(struct student \*s)

{

struct student \*temp,\*cur;

if(s==NULL)

printf("\n list is empty. Try option 1.\n");

else

{

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\n enter student details:\n");

printf("\n enter roll no:");

scanf("%d",&temp->rollno);

printf("\n enter name:");

fflush(stdin);

gets(temp->name);

printf("\n enter marks:");

scanf("%d",&temp->mark);

cur=s;

while(cur->next!=NULL)

cur=cur->next;

cur->next=temp;

temp->prev=cur;

}

}

void display(struct student \*s)

{

struct student \*temp,\*cur;

if(s==NULL)

printf("\n list is empty");

else

{

while(cur!=NULL)

{

printf("\nstudent details:\n");

printf("\nroll no:%d",cur->rollno);

printf("\nname:%s",cur->name);

printf("\nmarks:%d",cur->mark);

cur=cur->next;

}

}

}

struct student \*del\_at\_beg(struct student \*s)

{

struct student \*temp;

if(s==NULL)

printf("\nList is empty");

else

{

temp=s;

s=temp->next;

s->prev=NULL;

free(temp);

printf("\n first node deleted");

}

return s;

}

void del\_at\_end(struct student \*s)

{

struct student \*cur;

if(s==NULL)

printf("\n list is empty");

else

{

cur=s;

while(cur->next!=NULL)

{

prev=cur;

cur=cur->next;

}

prev->next=NULL;

free(cur);

}

}

void ins\_in\_btw(struct student \*s)

{

int rn;

struct student \*temp,\*cur,\*node;

if(s==NULL)

printf("\nlist is empty. Try option 1.\n");

else

{

printf("enter roll number to be searched for:");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

temp=(struct student \*)calloc(1,sizeof(struct student));

printf("\nenter roll no.:");

scanf("%d",&temp->rollno);

printf("\nenter name:");

fflush(stdin);

gets(temp->name);

temp->next=cur->next;

temp->prev=cur;

cur->next=temp;

node->prev=temp;

break;

}

else

{

cur=cur->next;

node=cur->next;

}

}

}

if(cur==NULL)

printf("\n roll no. not found");

}

void del\_in\_btw(struct student \*s)

{

int rn;

struct student \*temp,\*cur;

if(s==NULL)

printf("\nlist is empty");

else

{

printf("enter roll number to be searched for:");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

temp->next=cur->next;

free(cur);

break;

}

else

{

temp = cur;

cur=cur->next;

}

}

}

if(cur==NULL)

printf("\n roll no. not found");

}

void search(struct student \*s)

{

int rn;

struct student \*cur;

if(s==NULL)

printf("\n list is empty");

else

{

printf("\n enter roll no. to be searched for");

scanf("%d",&rn);

cur=s;

while(cur!=NULL)

{

if(cur->rollno==rn)

{

printf("\nstudent details:\n");

printf("\nroll no:%d",cur->rollno);

printf("\nname:%s",cur->name);

printf("\nmarks:%d",cur->mark);

}

cur=cur->next;

}

if(cur==NULL)

printf("\n roll no. not found");

}

}

void main()

{

int ch;

struct student \*start=NULL;

clrscr();

do

{

printf("\nChoose\n1. Insertion in beginning ");

printf("\n2. Insertion at end\n3. Insertion in between");

printf("\n4. Display\n5. Deletion in beginning");

printf("\n6. Deletion in end\n7. Deletion in between");

printf("\n8. Search\n9. Exit");

scanf("%d",&ch);

switch(ch)

{

case 1:

start=ins\_at\_beg(start);

break;

case 2:

ins\_at\_end(start);

break;

case 3:

ins\_in\_btw(start);

break;

case 4:

display(start);

break;

case 5:

start=del\_at\_beg(start);

break;

case 6:

del\_at\_end(start);

break;

case 7:

del\_in\_btw(start);

break;

case 8:

search(start);

break;

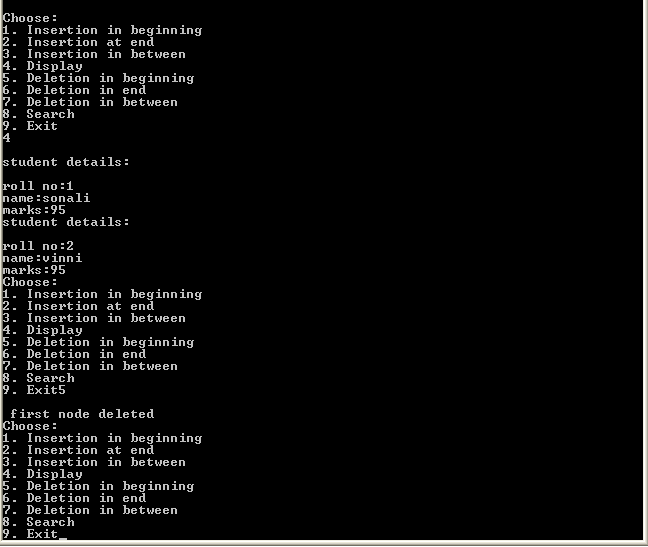
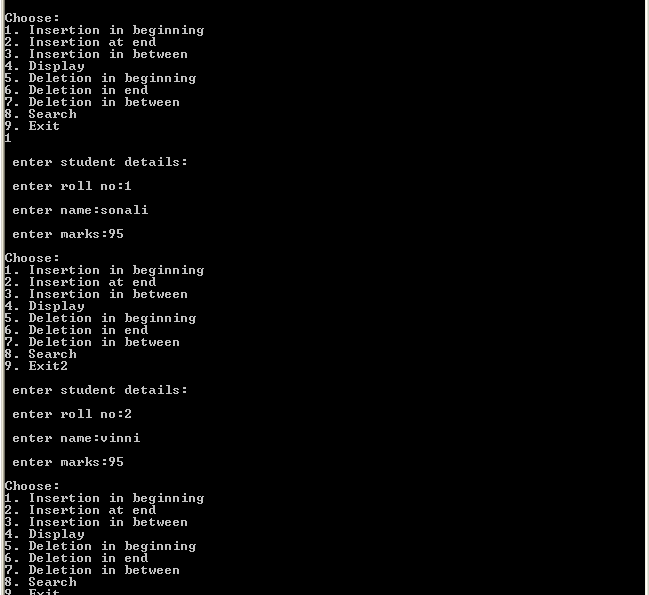
}

}while(ch<8);

getch();

}

Output:



**Experiment 6:**

**Write a C program that uses functions to perform the following operations on Binary Tree:**

**i) Creation ii) Insertion iii) Deletion**

Coding:

#include<stdio.h>

#include<conio.h>

typedef struct node

{

struct node \*left;

struct node \*right;

int data;

}bst;

bst \*root,\*queue[30];

int front=-1,rear=-1;

bst \*alloc(int n)

{

bst \*t;

t=(bst \*)calloc(1,sizeof(bst));

t->data=n;

return t;

}

void ins\_node(int n)

{

bst \*t,\*cur=root,\*prev;

t=alloc(n);

if(root==NULL)

root=t;

else

{

while(cur!=NULL)

{

prev=cur;

if(n<=cur->data)

cur=cur->left;

if(n>cur->data)

cur=cur->right;

}

if(n>prev->data)

prev->right=t;

if(n<prev->data)

prev->left=t;

}

}

void build()

{

int n;

do

{

printf("Enter data to insert.\n to stop enter 0.\n");

scanf("%d",&n);

if(n!=0)

ins\_node(n);

}while(n!=0);

}

void search(int n)

{

bst \*cur=root,\*t;

int fl=0;

if(root==NULL)

printf("Empty Tree.\n");

else

{

while(cur!=NULL)

{

if(n==cur->data)

{

printf("Data Found.\n");

fl=1;

break;

}

else

{

if(n<cur->data)

cur=cur->left;

if(n<cur->data)

cur=cur->right;

}

}

if(fl==0)

printf("Data Not Found.\n");

}

}

void del\_node(int n)

{

bst \*cur=root,\*t,\*prev,\*succ,\*temp;

int fl=0;

if(root==NULL)

printf("Empty Tree.\n");

else

{

while(cur!=NULL)

{

if(n==cur->data)

{

printf("Data Found.\n");

fl=1;

break;

}

else

{

if(n<cur->data)

cur=cur->left;

if(n<cur->data)

cur=cur->right;

}

}

if(fl==1)

{

if(cur->left==NULL && cur->right==NULL)

{

if(prev->left==cur)

prev->left=NULL;

if(prev->right==cur)

prev->right=NULL;

free(cur);

}

else if(cur->left!=NULL && cur->right==NULL)

{

if(prev->left==cur)

prev->left=cur->left;

if(prev->right==cur)

prev->right=cur->left;

free(cur);

}

else if(cur->right!=NULL && cur->left==NULL)

{

if(prev->left==cur)

prev->left=cur->right;

if(prev->right==cur)

prev->right=cur->right;

free(cur);

}

else if(cur->left!=NULL && cur->right!=NULL)

{

temp=cur;

succ=cur->right;

while(succ->left!=NULL)

{

prev=succ;

succ=succ->left;

}

temp->data=succ->data;

prev->left=NULL;

free(succ);

}

}

else

printf("Node Not Found.\n");

}

}

void enq(bst \*b)

{

if(front==-1)

front=0;

rear++;

queue[rear]=b;

}

bst \*deq()

{

bst \*t;

if(front>rear)

return NULL;

else

{

t=queue[front];

queue[front]=NULL;

front++;

return t;

}

}

void disp()

{

bst \*t=root,\*prev;

front=-1;

rear=-1;

while(front<=rear)

{

enq(t->left);

enq(t->right);

if(t!=NULL)

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

else

printf("\*");

prev=t;

t=deq();

if(t==NULL)

break;

if(t->data<prev->data)

printf("\n");

}

}

void main()

{

int cho,n;

clrscr();

do

{

printf("\nChoose:\n 1. Insert\n 2. Delete\n 3. Display\n 4. Search\n 5. Exit\n");

scanf("%d",&cho);

switch(cho)

{

case 1:

build();

break;

case 2:

printf("Enter data to delete.\n");

scanf("%d",&n);

del\_node(n);

break;

case 3:

disp();

break;

case 4:

printf("Enter data to search.\n");

scanf("%d",&n);

search(n);

break;

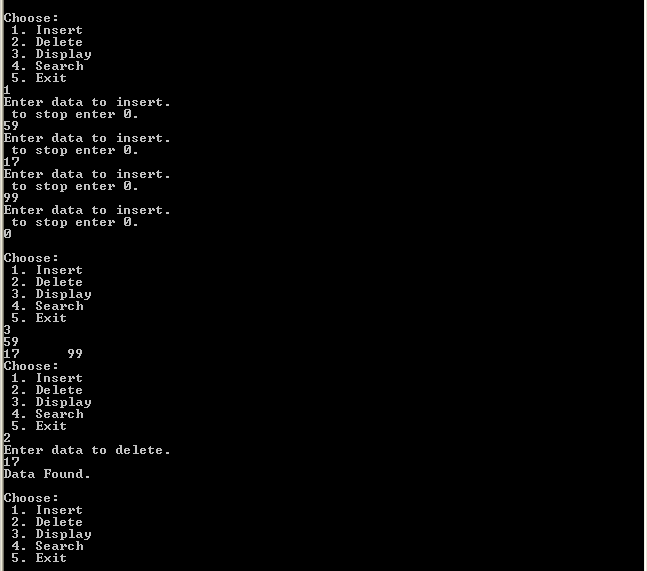
}

}while(cho<5);

getch();

}

Output:



**Experiment 7:**

**Write a C program that uses both recursive and non-recursive function to perform the Linear Search for a key value in a given list of integers.**

Coding:

# include <stdio.h>

void main()

{

int a[10], n=5, i, ele;

int linear(int[],int,int);

int r\_linear(int [],int, int);

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

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

{

printf("\n a[%d]\n", i+1);

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

}

printf("\n Search using simple function\n Element to find :");

scanf("%d", &ele);

i = linear(a,n, ele);

if(i == -1)

printf("\n Element not found \n");

else

printf("\n Element found at location %d\n", i);

printf("\n Search using recursive function \n Element to find :");

scanf("%d", &ele);

i = r\_linear(a,n-1, ele);

if(i == -1)

printf("\n Element not found ");

else

printf("\n Element found at location %d", i);

}

int linear(int a[], int n, int ele)

{

int i;

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

{

if (a[i] == ele)

return i+1;

}return -1;

}

int r\_linear(int a[], int n, int ele)

{

printf("%d",n);

if(n)

{

if (a[n] == ele)

return n+1;

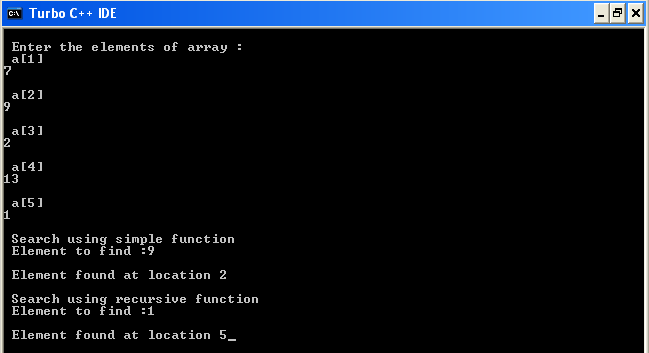
else

r\_linear(a, n-1, ele);

}return -1;

}

Output:



**Experiment 8:**

**Write a C program that uses both recursive and non-recursive function to perform Binary Search for the key value in the given list of integers.**

Coding:

# include <stdio.h>

void main(){

int a[10], n=5, i, ele,l=0;

int bin\_srch(int[],int,int, int);

int r\_bin\_srch(int [],int, int, int);

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

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

printf("\n a[%d]\n", i+1);

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

}

printf("\n BINARY Search using simple function\n Element to find :\n ");

scanf("%d", &ele);

i = bin\_srch(a,l,n-1,ele);

if(i == -1)

printf("\n Element not found \n");

else

printf("\n Element found at location %d\n", i);

printf("\n Binary Search using recursive function \n Element to find :\n ");

scanf("%d", &ele);

i = r\_bin\_srch(a,l,n-1,ele);

if(i == -1)

printf("\n Element not found ");

else

printf("\n Element found at location %d", i); }

int bin\_srch(int a[], int l, int u, int ele){

int mid;

while(l <= u){

mid = (l + u) /2;

if (a[mid] == ele)

return mid+1;

else if(a[mid] < ele)

l = mid +1;

else

u = mid -1;

}return -1;}

int r\_bin\_srch(int a[],int l,int u,int ele){

int mid;

if(l <= u){

mid = (l + u)/2;

if (a[mid] == ele)

return mid+1;

else if (a[mid] < ele)

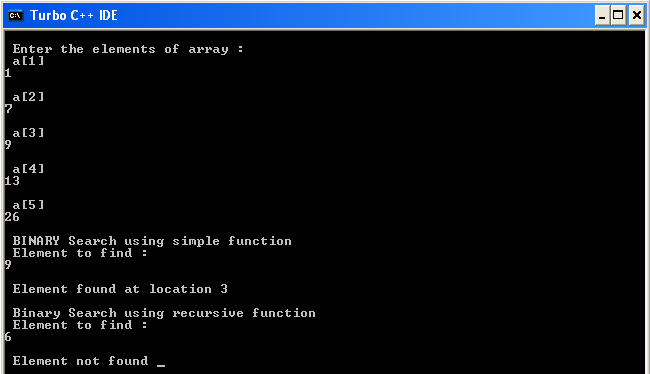
r\_bin\_srch(a,mid +1,u, ele);

else

r\_bin\_srch(a, l, mid -1, ele);

}return -1;}

Output:

****

**Experiment 9:**

**Write a C program that implement Bubble Sort method, insertion sort, selection sort method to sort a given list of integers in ascending or descending order.**

**Bubble Sort:**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,j,n=5,k=0,temp;

int a[5];

clrscr();

printf("Enter data.");

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

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

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

{

for(j=0;j<n-k;j++)

{

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

{

temp=a[j+1];

a[j+1]=a[j];

a[j]=temp;

}

}

k++;

}

printf(“\nSorted Order:\n\n”);

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

printf(“%d \t”,a[i]);

getch();

}

Output:



**Selection Sort:**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,j,pos=0,t,n=5;

int a[5];

clrscr();

printf(“Enter data.\n”);

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

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

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

{

pos=i;

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

{

if(a[j]<a[pos])

pos=j;

}

t=a[pos];

a[pos]=a[i];

a[i]=t;

}

printf(“\n Sorted Order:\n\n”);

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

printf(“%d \t”a[i]);

getch();

}

Output:



**Insertion Sort:**

Coding:

#include<stdio.h>

#include<conio.h>

void main()

{

int i,j,t,ele,n=5;

int a[5];

clrscr();

printf(“Enter data.\n”);

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

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

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

{

ele=a[i+1];

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

{

if(ele<a[j])

a[j+1]=a[j];

if(ele>a[j])

break;

}

a[j+1]=ele;

}

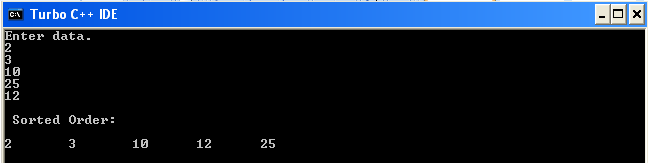
for(i=0;i<n;i++)

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

getch();

}

Output:



**Experiment 10:**

**Write a C program that implements Quick Sort method to sort a given list of integers in ascending order.**

Coding:

#include<stdio.h>

void quicksort(int [10],int,int);

void main()

{

int x[20],size,i;

printf("Enter size of the array: \n");

scanf("%d",&size);

printf("Enter %d elements: \n",size);

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

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

quicksort(x,0,size-1);

printf("Sorted elements: \n");

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

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

}

void quicksort(int x[10],int first,int last)

{

int pivot,j,temp,i;

if(first<last)

{

pivot=first;

i=first;

j=last;

while(i<j)

{

while(x[i]<=x[pivot]&&i<last)

i++;

while(x[j]>x[pivot])

j--;

if(i<j)

{

temp=x[i];

x[i]=x[j];

x[j]=temp;

}

}

temp=x[pivot];

x[pivot]=x[j];

x[j]=temp;

quicksort(x,first,j-1);

quicksort(x,j+1,last);

}

}

Output:

