**LAB PROGRAM 1:** **(SEARCH IN SPARSE MATRIX)🡪Design, develop, and execute a program in C to read a sparse matrix of integer values and to search the sparse matrix for an element specified by the user. Print the result of the search appropriately. Use the triple <row, column, value> to represent an element in the sparse matrix.**

#include<stdio.h>

#include<conio.h>

#define MAX\_TERMS 101

int k;

typedef struct

{

int row,col,val;

}TERM;

//FUNCTION TO READ SPARSE MATRIX AS A TRIPLE

void read\_sparse\_matrix(TERM a[],int m,int n)

{

int i,j,item;

a[0].row=m;

a[0].col=n;

k=1;

printf("Enter the elements\n");

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

{

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

{

scanf("%d",&item);

if(item==0) continue;

a[k].row=i;

a[k].col=j;

a[k].val=item;

printf("Non zero element is stored in location a[%d].val=%d\n",k,a[k].val);

k++;

}

}

a[0].val=k-1;

}

void print\_sparse\_matrix(TERM a[])

{

int p;

printf("Non zero elements are present in the following location \n");

for(p=1;p<k;p++)

{

printf("row=%d col=%d val=%d\n",a[p].row,a[p].col,a[p].val);

}

}

//FUNCTION TO SEARCH FOR AN ITEM IN SPARSE MATRIX

void search(TERM a[],int item)

{

int i,j;

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

{

if(item==a[i].val)

{

printf("Search is successful, Element found at pos %d\n",i);

getch();

exit(0);

}

}

printf("Search is unsuccessful\n");

}

void main()

{

int m,n,item;

TERM a[MAX\_TERMS];

clrscr();

printf("Enter the number of Rows & Columns\n");

scanf("%d %d",&m,&n);

read\_sparse\_matrix(a,m,n);

print\_sparse\_matrix(a);

printf("Enter the element to be searched\n");

scanf("%d",&item);

search(a,item);

getch();

}

**OUTPUT**

**Enter the number of Rows & Columns**

**3 3**

**Enter the elements**

**1**

**Non zero element is stored in location a[1].val=1**

**0**

**0**

**0**

**2**

**Non zero element is stored in location a[2].val=2**

**4**

**Non zero element is stored in location a[3].val=4**

**0**

**0**

**0**

**Non zero elements are present in the following location**

**row=0 col=0 val=1**

**row=1 col=1 val=2**

**row=1 col=2 val=4**

**Enter the element to be searched**

**4**

**Search is successful, Element found at pos 3**

**LAB PROGRAM 13:** **(RECURSION)🡪Write recursive C Programs for**

* 1. **Searching an element on a given list of integers using the Binary Search method.**

#include<stdio.h>

#include<conio.h>

//FUNCTION TO SEARCH AN ELEMENT USING BINARY SEARCH LOGIC

int search(int key,int a[],int low,int high)

{

int mid;

if(low>high) return -1;

mid=(low+high)/2;

if(key==a[mid])

return mid;

if(key<a[mid])

search(key,a,low,mid-1);

else

search(key,a,mid+1,high);

}

void main()

{

int n,i,a[20],key,pos;

clrscr();

printf("enter the number of elements\n");

scanf("%d",&n);

printf("Enter the elements\n");

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

{

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

}

printf("Enter the key ele to be searched\n");

scanf("%d",&key);

pos=search(key,a,0,n-1);

if(pos==-1)

printf("Key ele not found\n");

else

printf("Ele found at pos %d\n",pos);

getch();

}

**OUTPUT**

**enter the number of elements**

**5**

**Enter the elements**

**5**

**6**

**7**

**8**

**9**

**Enter the key ele to be searched**

**7**

**Ele found at pos 2**

**b. Solving the Towers of Hanoi problem.**

#include<stdio.h>

#include<conio.h>

void tower(int n,char src,char temp,char des)

{

if(n==1)

{

printf("move disc %d from %c to %c\n",n,src,des);

return;

}

tower(n-1,src,des,temp);

printf("move disc %d from %c to %c\n",n,src,des);

tower(n-1,temp,src,des);

}

void main()

{

int n;

clrscr();

printf("Enter the number of discs\n");

scanf("%d",&n);

tower(n,'S','T','D');

getch();

}

**OUTPUT1**

**Enter the number of discs**

**3**

**move disc 1 from S to D**

**move disc 2 from S to T**

**move disc 1 from D to T**

**move disc 3 from S to D**

**move disc 1 from T to S**

**move disc 2 from T to D**

**move disc 1 from S to D**

**OUTPUT2**

**Enter the number of discs**

**4**

**move disc 1 from S to T**

**move disc 2 from S to D**

**move disc 1 from T to D**

**move disc 3 from S to T**

**move disc 1 from D to S**

**move disc 2 from D to T**

**move disc 1 from S to T**

**move disc 4 from S to D**

**move disc 1 from T to D**

**move disc 2 from T to S move disc 1 from D to S**

**move disc 3 from T to D move disc 1 from S to T**

**move disc 2 from S to D move disc 1 from T to D**