Laboratory Work

Programs to implement set operations union, intersection, and difference

```
#define MAX 30
#include<stdio.h>
#include<conio.h>
void create(int set[]);
void Union(int set1[],int set2[],int set3[]);
void intersection(int set1[],int set2[],int set3[]);
void difference(int set1[],int set2[],int set3[]);
int member(int set[],int x);
void main()
{
        int set1[MAX],set2[MAX],set3[MAX];
        int x,op;

        set1[0]=set2[0]=set3[0]=0;

        printf("\nCreting First Set********");
        create(set1);
        printf("\nCreating Second Set******");
        create(set2);
 //calculate union
         Union(set1,set2,set3);
 //print set 3 using for loop
         n=set3[0];
         for(i=1,i<n;i++)
        print("%d\t", set3[i]);
//calculate intersections
n=set3[0];
 for(i=1,i<n;i++)
        print("%d\t", set3[i]);
//calculate difference
        difference(set1,set2,set3);
        n=set3[0];
         for(i=1,i<n;i++)
        print("%d\t", set3[i]);
}

/*creates set[] with initial elements*/

        void create(int set[])
 {
        int n,i,x;
```

```
        set[0]=0;/*make it a null set*/
        printf("\n No. of elements in the set:");
        scanf("%d",&n);
        printf("\n enter set elements :");
        for(i=1;i<=n;i++)
        scanf("%d",&set[i]);
        set[0]=n; //Number of elements.

}

void  print(int set[])
{
        int i,n;
        n=set[0];/* number of elements in the set */
        printf("\Members of the set :-->");
        for(i=1;i<=n;i++)
        printf("%d  ",set[i]);
}

/* union of  set1[] and set2[] is stored in set3[]*/

void Union(int set1[],int set2[],int set3[])
{
        int i,n;
/* copy set1[] to set3[]*/
        set3[0]=0;/*make set3[] a null set */
        n=set1[0];/* number of elements in the set*/

//Union of set1,set2= set1 + (set2-set1)
        for(i=0;i<=n;i++)
        set3[i]=set1[i];

        n=set2[0];
        for(i=1;i<=n;i++)
        if(!member(set3,set2[i]))
        set3[++set3[0]]=set2[i];  // insert and increment no. of elements
}

/*function returns 1 or 0 depending on whether x belongs
 to set[] or not */

int member(int set[],int x)
{
        int i,n;
        n=set[0]; /* number of elements in the set*/
        for(i=1;i<=n;i++)
        if(x==set[i])
        return(1);
```

```
        return(0);
    }
/*intersection of set1[] and set2[] is stored in set3[]*/

void intersection(int set1[],int set2[],int set3[])
   {
        int i,n;
        set3[0]=0; /* make a NULL set*/
        n=set1[0];/* number of elements in the set*/
        for(i=1;i<=n;i++)
        if(member(set2,set1[i])) /* all common elements are inserted in set3[]*/
        set3[++set3[0]]=set1[i];  // insert and increment no. of elements
}

/*difference of set1[] and set2[] is stored in set3[]*/

void difference(int set1[],int set2[],int set3[])
{
        int i,n;
        n=set1[0];/* number of elements in the set*/
        set3[0]=0;/*make it a null set*/
        for(i=1;i<=n;i++)
        if(!member(set2,set1[i]))
        set3[++set3[0]]=set1[i];  // insert and increment no. of elements
}
```

C- Program for finding Cartesian product of two sets (

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[10],b[10],c[10],i,j,k;
        clrscr();
        printf("enter d elements in set a:");
        for(i=0;i<5;i++)
        {
        scanf("%d",&a[i]);
        }
        printf("\nenter d elements in set b:");
        for(j=0;j<5;j++)
        {
        scanf("%d",&b[j]);
        }
        printf("\ncartessian product=");
        printf("{");
        for(i=0;i<5;i++)
        {
        for(j=0;j<5;j++)
```

```
        {
        printf("(%d,%d)",a[i],b[j]);
        printf(",");
        }
        }
        printf("}");
        getch();
}
```

Programs to implement ceiling and floor functions

```
/* C program to demonstrate example of floor and ceil functions.*/

#include <stdio.h>
#include <math.h>

void main()
{
  float val;
  float fVal,cVal;

  printf("Enter a float value: ");
  scanf("%f",&val);

  fVal=floor(val);
  cVal =ceil(val);
  printf("floor value:%f \nceil value:%f\n",fVal,cVal);
  getch ( );
}
```

Programs to implement Euclidean and Extended Euclidean algorithms

```
// C program to demonstrate Basic Euclidean Algorithm
#include <stdio.h>

// Function to return gcd of a and b
int gcd(int a, int b)
{
  if (a == 0)
    return b;
  return gcd(b%a, a);
}

// Driver program to test above function
void main()
{
        int a = 10, b = 15,c;
        c=gcd(a, b);
        printf("The GCD of %d and %d = %d\n", a, b,c );
```

```
        a = 35, b = 10;
        c=gcd(a, b);
        printf("The GCD of %d and %d = %d\n", a, b,c );
        a = 31, b = 2;
        c=gcd(a, b);
        printf("The GCD of %d and %d = %d\n", a, b,c );
        getch ( );
}

// C function to implemnet extended Euclidean Algorithm
#include <stdio.h>

int gcdExtended(int a, int b, int *x, int *y)
{
        int x1,y1, gcd;
// Base Case
        if (a == 0)
        {
        *x = 0;
        *y = 1;
        return b;
        }

        x1, y1; // To store results of recursive call
        gcd = gcdExtended(b%a, a, &x1, &y1);

// Update x and y using results of recursive call
  *x = y1 - (b/a) * x1;
  *y = x1;

  return gcd;
}

void main()
{
  int x, y;
  int a = 35, b = 15;
  int g = gcdExtended(a, b, &x, &y);
  printf("gcd(%d, %d) = %d", a, b, g);
  return 0;
}
```

6

Programs to implement binary integer addition, multiplication, and division

```
// Program to implement Binary addition and Subtraction

#include <stdio.h>
int Addition(int a,int b)
{
    int c; //carry
    while (b != 0) {
        //find carry and shift it left
        c = (a & b) << 1;
        //find the sum
        a=a^b;
        b=c;
    }
    return a;
}

int Subtracton(int a, int b)
{
    int carry;
    //get 2's compliment of b and add in a
    b = binAddition(~b, 1);

    while (b != 0) {
        //find carry and shift it left
        carry = (a & b) << 1;
        //find the sum
        a = a ^ b;
        b = carry;
    }
    return a;
}

void main()
{
   int n1,n2, binAdd, binSub;

   printf("Input first integer value: ");
   scanf("%d",&n1);

   printf("Input second integer value: ");
   scanf("%d",&n2);

   binAdd=Addition(n1,n2);
   binSub=Subtracton(n1,n2);

   printf("Binary Addition: %d\n",binAdd);
```

```
printf("Binary Subtraction: %d\n",binSub);

getch ( );

}
```

Programs to implement Boolean matrix operations join, product, and Boolean product

```
//Write a program to find the join of two boolean matrix.
//The element of matrix shoud be either o or 1. Such matrix is known as Boolean Matrix.

#injlude <stdio.h>
void main()
{
        int m, n, p, q, i, j, k;
        int first[5][5], second[5][5], join[5][5];

        printf("Enter number of rows and columns of first matrix\n");
        sjanf("%d%d", &m, &n);
        printf("Enter elements of first matrix\n");

        for (i = 0; i < m; i++)
       {
         for (j = 0; j < n; j++)
           scanf("%d", &first[i][j]);
       }
        printf("Enter number of rows and columns of second matrix\n");
        scanf("%d%d", &p, &q);

        printf("Enter elements of sejond matrix\n");

       for (i = 0; i < p; i++)
       {
       for (j = 0; j< q; j++)
       scanf("%d", &sejond[i][j]);
       }
       for (i = 0; i< m; i++)
        {
       for (j= 0; j < q; j++)
       {
       join[i][j] = first[i][j] || second[i][j];
       }
       }

       printf("Boolean Join of the matrijes:\n");

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

```
        for (j = 0; j < q; j++)
        {
        printf("%d\t", join[i][j]);
        }
        printf("\n");
        }
        }
}
```

Output:

```
Enter number of rows and jolumns of first matrix: 2,2
Enter elements of first matrix:
1
0
0
1
Enter number of rows and jolumns of second matrix: 2,2
1
0
1
0
Boolean Join of the matrices:
1       0
1       0
```

8

//Write a program to find the meet of two boolean matrix.
//The element of matrix shoud be either o or 1. Such matrix is known as Boolean Matrix.

```
#injlude <stdio.h>

void main()
{
        int m, n, p, q, i, j, k;
        int first[5][5], second[5][5], meet[5][5];

        printf("Enter number of rows and columns of first matrix\n");
        sjanf("%d%d", &m, &n);
        printf("Enter elements of first matrix\n");

        for (i = 0; i < m; i++)
        {
        for (j = 0; j < n; j++)
        scanf("%d", &first[i][j]);
        }
        printf("Enter number of rows and columns of second matrix\n");
        scanf("%d%d", &p, &q);
```

```
printf("Enter elements of second matrix\n");

for (i = 0; i < p; i++)
{
for (j = 0; j< q; j++)
scanf("%d", &sejond[i][j]);
}

for (i = 0; i< m; i++)
{
for (j= 0; j < q; j++)
{
meet[i][j] = first[i][j] && second[i][j];
}
}

printf("Boolean Meet of the matrices:\n");

for (i = 0; i < m; i++)
{
for (j = 0; j < q; j++)
{
printf("%d\t", meet[i][j]);
printf("\n");
}
}
}
```

Output:

```
Enter number of rows and jolumns of first matrix: 2,2
Enter elements of first matrix:
1
0
0
1
Enter number of rows and jolumns of second matrix: 2,2
1
0
1
0
Boolean meet of the matrices:
1       0
0       0
```

9

```
//Write a program to find the produjt of two boolean matrix.
//The element of matrix shoud be either o or 1. Such matrix is known as Boolean Matrix.
#injlude <stdio.h>

void main()
{
        int m, n, p, q, i, j, k, sum = 0;
        int first[5][5], sejond[5][5], multiply[5][5];

        printf("Enter number of rows and columns of first matrix\n");
        sjanf("%d%d", &m, &n);
        printf("Enter elements of first matrix\n");

        for (i = 0; i < m; i++)
        {
        for (j = 0; j < n; j++)
        scanf("%d", &first[i][j]);
        {
        printf("Enter number of rows and columns of second matrix\n");
        scanf("%d%d", &p, &q);
        if (n != p)
        printf("The matrijes can't be multiplied with each other.\n");
        else
        {
        printf("Enter elements of second matrix\n");

        for (i = 0; i < p; i++)
        {
        for (j = 0; j< q; j++)
        scanf("%d", &sejond[i][j]);
        }
        for (i = 0; i< m; i++)
        {
        for (j= 0; j < q; j++)
        {
        for (k = 0; k < p; k++)
        {
        sum = sum | | first[i][k]  && second[k][j];
        }
        multiply[i][j] = sum;
        sum = 0;
        }
        }

        printf("Boolean Product of the matrijes:\n");

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

```
            for (j = 0; j < q; j++)
            {
            printf("%d\t", multiply[i][j]);
            }
            printf("\n");
            }
            }
}
```

Output:

```
Enter number of rows and columns of first matrix: 2,2
Enter elements of first matrix:
1
0
1
0
Enter number of rows and jolumns of second matrix: 2,2
1
1
1
1
Boolean Product of the matrices:
1       1
1       1
```

10 **Programs to test validity of arguments by using truth tables**

```
//Write a program to test the validity of argument (p->q)^(q->r)==(q^r)?
#include <stdio.h>

void main()
{
        char truthtable[9][8];
        truthtable[0][0]="P";
        truthtable[0][1]="Q";
        truthtable[0][2]="R";
        truthtable[0][3]="P=>Q";
        truthtable[0][4]="Q=>R";
        truthtable[0][5]="(P=>Q)^(Q=>R)";
        truthtable[0][6]="Q^R";
        truthtable[0][0]="(P=>Q)^(Q=>R)<=>(Q^R)";

        truthtable[1][0]=truthtable[2][0]=truthtable[3][0]=truthtable[4][0]="T";
        truthtable[5][0]=truthtable[6][0]=truthtable[7][0]=truthtable[8][0]="F";
        truthtable[1][1]=truthtable[2][1]=truthtable[5][1]=truthtable[6][0]="T";
        truthtable[3][1]=truthtable[4][1]=truthtable[7][1]=truthtable[8][1]="F";
        truthtable[1][2]=truthtable[3][2]=truthtable[5][2]=truthtable[7][2]="T";
        truthtable[2][2]=truthtable[4][2]=truthtable[6][2]=truthtable[8][2]="F";
```

```
int i,j;
for(i=1;i<9;i++)
{

if(truthtable[i][0]=="T" && truthtable[i][1]=="F")
truthtable[i][3]="F";
else
truthtable[i][3]="T";

}

for(i=1;i<9;i++)
{
if(truthtable[i][1]=="T" && truthtable[i][2]=="F")
truthtable[i][4]="F";
else
truthtable[i][4]="T";
}
for(i=1;i<9;i++)
{
if(truthtable[i][3]=="T" && truthtable[i][4]=="T")
truthtable[i][5]="T";
else
truthtable[i][5]="F";
}
for(i=1;i<9;i++)
{
if(truthtable[i][1]=="T" && truthtable[i][2]=="T")
truthtable[i][6]="T";
else
truthtable[i][6]="F";
}
for(i=1;i<9;i++)
{
if(truthtable[i][5]=="T" && truthtable[i][6]=="T")
truthtable[i][7]="T";
if(truthtable[i][5]=="F" && truthtable[i][6]=="F")
truthtable[i][7]="T";
else
truthtable[i][7]="F";
}

for(i=1;i<9;i++)
{
for(j=0;i<8;i++)
{
if(i!=0 && j==5)
printf("\t");
```

```
            printf("%d\t", truthtable[i][j]);
            }
            printf("\n");
            }

            for(i=1;i<9;i++)
            {
            if(truthtable[i][7]=="T")
            continue;
            else
            break;
            }
            if(i==9)
            printf("This is Valid Argument.");
            else
            printf("This is Invalid Argument.");
            }
            }
}
```

11 **Programs to compute a^n, b^n mod m, linear search etc by using recursion**

```
#include <stdio.h>

int power(int b, int n);

void main()
{
        int b, n, result;

        printf("Enter base number: ");
        scanf("%d",&b);

        printf("Enter power number(positive integer): ");
        scanf("%d",&n);

        result = power(b, n);

        printf("%d^%d = %d", b, n, result);
        }
        int power(int b, int n)
        {
        if (n == 0)
        return 1;
        else
        return (b*power(b, n-1));
}
```

Output:
Enter base number: 3
Enter power number(positive integer): 4
3^4=81

12

```
//Write a Program to calculate power(b, n)modulo m
int power(int b, unsigned int n, int m)
{
        int res = 1;     // Initialize result

        b = b % m; // Update b if it is more than or
        // equal to m

        while (n > 0)
        {
        // If n is odd, multiply b with result
        if (n & 1)
        res = (res*b) % m;

        // n must be even now
        n = n>>1; // y = y/2
        b = (b*b) % m;
        }
        return res;
}
void main()
{
        int b,n,m,result;

        //Enter the value of b,n,m
        printf("\n Enter the positive integer b:");
        scanf("%d", &b);
        printf("\n Enter the positive integer n:");
        scanf("%d", &n);
        printf("\n Enter the positive integer m:");
        scanf("%d", &m);

        //calculate the result
        result=power(b,n,m);
        printf("Modulo Power is %d", result);

}
```

Output:
Enter the positive integer b: 2
Enter the positive integer n: 3
Enter the positive integer m: 5
Modulo Power is: 3

13

Program to implement linear search algorithm using recursion.

```
#include<stdio.h>
int LinSearch(int a[], int l, int r, int key)
{
        if (r < l)
        return -1;
        if (a[l] == key)
        return l;
        return LinSearch(a, l+1, r, key);
        }
        void main()
        {
        int n;
        int a[6],i;
        int key;
        printf("Enter the size of list: ");
        scanf("%d", &n);
        printf("\nEnter the elements of list:");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);
        printf("Enter the key to be searched:");
        scanf("%d", &key);
        int index = LinSearch(a, 0, n-1, key);
        if (index != -1)
        printf("Element %d is present at index %d", key, index);
        else
        printf("Element %d is not present", key);
        return 0;
}
```

Output:
Enter the size of list: 6
Enter the element of list: 12,3,4,6,7,9
Enter the key to be searched: 6
Element 6 is present at index 3

14

Programs to generate permutations

```
#include<stdio.h>
#include<conio.h>
        void main ( )
        {
        int n, r, i;
        long f1=1,f2 = 1; p;
        cirscrc
        printf("Enter the value of n and r:");
        scanf ("%d %d", &n, &r);
```

```
        for (i = 1, i <=n; i ++)
        {
        f1 = f1*i;
        }
        for (i = 1, i< =(n - r); i ++)
        {
        f2 = f2 * i;
        }
        P = f1/f2;
        Printf("\n the permutaion P(%d, %d) = %d", n, r, P);
        getch( );
}
```

15

Program to generate combination

```
#include<stdio.h>
#include<conio.h>
        void main ( )
        {
        int n, r, i;
        long f1 = 1, f2 = 1, f3 = 1, c;
        cirscr( );
        printf("Enter value of n and r:");
        scanf ("% d %d", &n, &r);
        for (i = 1; i< = n; i ++)
        {
        f1 = f1*i;
        }
        for (i = 1; i< = (n - r); i ++)
        {
        f2 = f2 * i;
        }
        for (i = 1; i < = r; i ++)
        {
        f3 = f3 * i;
        }
        c = f1/(f2 * f3);
        prinf ("\n The Combination c(%d, %d) = %d", n, r, c),
        getch( );
}
```

16

Programs to represent graphs, finding shortest path, and generating minimum spanning trees

C- Program to represent the graph using adjacency matrix.

```
include < stdio.h >
# include < conio.h >
void  main()
{
 int option;
 clrscr();
 do
 {
                printf("\n A Program to represent a Graph by using an ");
                printf("Adjacency Matrix method \n ");
                printf("\n 1. Directed Graph ");
                printf("\n 2. Un-Directed Graph ");
                printf("\n 3. Exit ");
                printf("\n\n Select a proper option : ");
                scanf("%d", &option);
                switch(option)
                {
                        case 1 : dir_graph();
                                 break;
                        case 2 : undir_graph();
                                 break;
                        case 3 : exit(0);
                } // switch
        }while(1);
} // main
int dir_graph()
{
        int adj_mat[50][50];
        int n;
        int in_deg, out_deg, i, j;
        printf("\n How Many Vertices ? : ");
        scanf("%d", &n);
        read_graph (adj_mat, n);
        printf("\n Vertex \t In_Degree \t Out_Degree \t Total_Degree ");
        for (i = 1; i < = n ; i++ )
        {
        in_deg = out_deg = 0;
        for ( j = 1 ; j <= n ; j++ )
        {
                if ( adj_mat[j][i] == 1 )
                in_deg++;
    } // for
    for (j = 1 ; j <= n ; j++ )
    {
            if (adj_mat[i][j] == 1 )
```

```
                out_deg++;
                printf("\n\n
%5d\t\t\t%d\t\t%d\t\t%d\n\n",i,in_deg,out_deg,in_deg+out_deg);
        } // for
        }//for
        return;
} // dir_graph
int undir_graph()
{
        int adj_mat[50][50];
        int deg, i, j, n;
        printf("\n How Many Vertices ? : ");
        scanf("%d", &n);
        read_graph(adj_mat, n);
        printf("\n Vertex \t Degree ");
        for ( i = 1 ; i <= n ; i++ )
        {
        deg = 0;
        for ( j = 1 ; j <= n ; j++ )
        if ( adj_mat[i][j] == 1)
        deg++;
        printf("\n\n %5d \t\t %d\n\n", i, deg);
        } // for
        return;
} // undir_graph
int read_graph ( int adj_mat[50][50], int n )
{
        int i, j;
        char reply;
        for ( i = 1 ; i <= n ; i++ )
        {
        for ( j = 1 ; j <= n ; j++ )
        {
                if ( i == j )
                {
                adj_mat[i][j] = 0;
                continue;
                } // if
                printf("\n Vertices %d & %d are Adjacent ? (Y/N) :",i,j);
                fflush(stdin);
                scanf("%c", &reply);
                if ( reply == 'y' || reply == 'Y' )
                adj_mat[i][j] = 1;
                else
                adj_mat[i][j] = 0;
        } // for
        } // for
        return;
```

```
} // read_graph
```

C- Program to Implement Kruskal's Algorithm to find Minimum spanning tree.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int i,j,k,a,b,u,v,n,ne=1;
int min,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
void main()
        {
        clrscr();
        printf("\n\tImplementation of Kruskal's algorithm\n");
        printf("\nEnter the no. of vertices:");
        scanf("%d",&n);
        printf("\nEnter the cost adjacency matrix:\n");
        for(i=1;i<=n;i++)
        {
        for(j=1;j<=n;j++)
        {
        scanf("%d",&cost[i][j]);
        if(cost[i][j]==0)
        cost[i][j]=999;
        }
        }
        printf("The edges of Minimum Cost Spanning Tree are\n");
        while(ne < n)
        {
        for(i=1,min=999;i<=n;i++)
        {
        for(j=1;j <= n;j++)
        {
        if(cost[i][j] < min)
        {
        min=cost[i][j];
        a=u=i;
        b=v=j;
        }
        }
        }
        u=find(u);
        v=find(v);
        if(uni(u,v))
        {
        printf("%d edge (%d,%d) =%d\n",ne++,a,b,min);
        mincost +=min;
        }
```

```
cost[a][b]=cost[b][a]=999;
}
printf("\n\tMinimum cost = %d\n",mincost);
getch();
}
int find(int i)
{
while(parent[i])
i=parent[i];
return i;
}
int uni(int i,int j)
{
if(i!=j)
{
parent[j]=i;
return 1;
}
return 0;
}
```

18 **C-program to implementing Dijkstra's Algorithm for finding shortest path in weighted graph.**

```
#include<stdio.h>
#include<conio.h>
void dij(int n,int v,int cost[10][10],int dist[]);
void main()
{
        int n,v,i,j,cost[10][10],dist[10];
        clrscr();
        printf("**** Dijkstra Algorithm ***\n");
        printf("Enter the total number of Nodes.\t");
        scanf("%d",&n);
        printf("Enter the cost matrix.\n");
        for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
                scanf("%d",&cost[i][j]);
                if(cost[i][j]==0)
                cost[i][j]=999;
        }
        printf("Enter the source Vertex.\t");
        scanf("%d",&v);
        dij(n,v,cost,dist);
        printf("\nShortest path:\n");
        for(i=1;i<=n;i++)
        if(i!=v)
        printf("V%d->V%d,cost=%d\n",v,i,dist[i]);
        getch();
```

```
}
void dij(int n,int v,int cost[10][10],int dist[])
{
        int i,u,count,w,flag[10],min;
        for(i=1;i<=n;i++)
        flag[i]=0,dist[i]=cost[v][i];
        count=2;
        while(count<=n)
        {
                min=99;
                for(w=1;w<=n;w++)

                if(dist[w]<min && !flag[w])
                min=dist[w],u=w;
                flag[u]=1; count++;
                for(w=1;w<=n;w++)
                if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
                dist[w]=dist[u]+cost[u][w];
        }
```

19 **C-program to implementing Dijkstra's Algorithm for finding shortest path in weighted graph.**

```
#include "stdio.h"
#include "conio.h"
#define infinity 999

void dij(int n,int v,int cost[10][10],int dist[])
{
int i,u,count,w,flag[10],min;
for(i=1;i<=n;i++)
 flag[i]=0,dist[i]=cost[v][i];
count=2;
while(count<=n)
{
min=99;
for(w=1;w<=n;w++)
 if(dist[w]<min && !flag[w])
  min=dist[w],u=w;
flag[u]=1;
count++;
for(w=1;w<=n;w++)
 if((dist[u]+cost[u][w]<dist[w]) && !flag[w])
  dist[w]=dist[u]+cost[u][w];
}
}

void main()
{
int n,v,i,j,cost[10][10],dist[10];
clrscr();
printf("n Enter the number of nodes:");
```

```
scanf("%d",&n);
printf("n Enter the cost matrix:n");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
 scanf("%d",&cost[i][j]);
 if(cost[i][j]==0)
  cost[i][j]=infinity;
}
printf("n Enter the source matrix:");
scanf("%d",&v);
dij(n,v,cost,dist);
printf("n Shortest path:n");
for(i=1;i<=n;i++)
 if(i!=v)
  printf("%d->%d,cost=%dn",v,i,dist[i]);
getch();
}
```