NAME:	Bhavesh Prashant Chaudhari
UID:	2021300018
SUBJECT	DAA
EXPERIMENT NO :	08
AIM:	To solve 15 puzzle problem using Branch and Bound.
PROBLEM STATEMENT 1:	
THEORY	Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in worst case. The Branch and Bound Algorithm technique solves these problems relatively quickly. Given a 4x4 board with 16 tiles (every tile has one number from 1 to 16) and one empty space. The objective is to place the numbers on tiles to match the final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.
PROGRAM:	see below

```
#include<stdio.h>

int m=0,n=4;

int cal(int temp[10][10],int t[10][10])
{
        int i,j,m=0;
        for(i=0;i < n;i++)
```

```
                for(j=0;j < n;j++)
                {
                        if(temp[i][j]!=t[i][j])
                        m++;
                }
        return m;
}

int check(int a[10][10],int t[10][10])
{
        int i,j,f=1;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        if(a[i][j]!=t[i][j])
                                f=0;
        return f;
}


void main()
{
        int p,i,j,n=4,a[10][10],t[10][10],temp[10][10],r[10][10];
        int m=0,x=0,y=0,d=1000,dmin=0,l=0;
        printf("\nEnter the matrix to be solved,space with zero
:\n");
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        scanf("%d",&a[i][j]);

        printf("\nEnter the target matrix,space with zero :\n");
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        scanf("%d",&t[i][j]);
```

```
printf("\nEntered Matrix is :\n");
for(i=0;i < n;i++)
{
        for(j=0;j < n;j++)
                printf("%d\t",a[i][j]);
        printf("\n");
}

printf("\nTarget Matrix is :\n");
for(i=0;i < n;i++)
{
        for(j=0;j < n;j++)
                printf("%d\t",t[i][j]);
        printf("\n");
}

while(!(check(a,t)))
{
        l++;
        d=1000;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                {
                        if(a[i][j]==0)
                        {
                                x=i;
                                y=j;
                        }
                }

        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        temp[i][j]=a[i][j];
```

```
if(x!=0)
{
    p=temp[x][y];
    temp[x][y]=temp[x-1][y];
    temp[x-1][y]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
    d=dmin;
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            r[i][j]=temp[i][j];
}

for(i=0;i < n;i++)
    for(j=0;j < n;j++)
        temp[i][j]=a[i][j];
if(x!=n-1)
{
    p=temp[x][y];
    temp[x][y]=temp[x+1][y];
    temp[x+1][y]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
    d=dmin;
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            r[i][j]=temp[i][j];
}
```

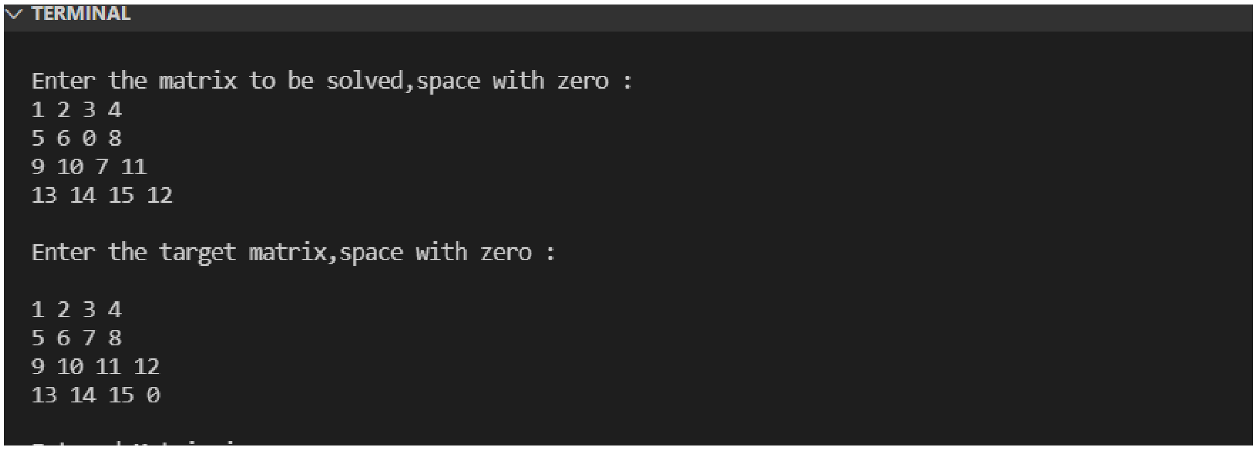
```
for(i=0;i < n;i++)
        for(j=0;j < n;j++)
                temp[i][j]=a[i][j];
if(y!=n-1)
{
        p=temp[x][y];
        temp[x][y]=temp[x][y+1];
        temp[x][y+1]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
        d=dmin;
        for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                        r[i][j]=temp[i][j];
}

for(i=0;i < n;i++)
        for(j=0;j < n;j++)
                temp[i][j]=a[i][j];
if(y!=0)
{
        p=temp[x][y];
        temp[x][y]=temp[x][y-1];
        temp[x][y-1]=p;
}
m=cal(temp,t);
dmin=l+m;
if(dmin < d)
{
        d=dmin;
```

```
                for(i=0;i < n;i++)
                    for(j=0;j < n;j++)
                        r[i][j]=temp[i][j];
            }

            printf("\nCalculated Intermediate Matrix Value
:\n");
            for(i=0;i < n;i++)
            {
                for(j=0;j < n;j++)
                  printf("%d\t",r[i][j]);
                printf("\n");
            }
            for(i=0;i < n;i++)
                for(j=0;j < n;j++)
                {
                  a[i][j]=r[i][j];
                  temp[i][j]=0;
                }
            printf("Minimum cost : %d\n",d);
        }
}
```

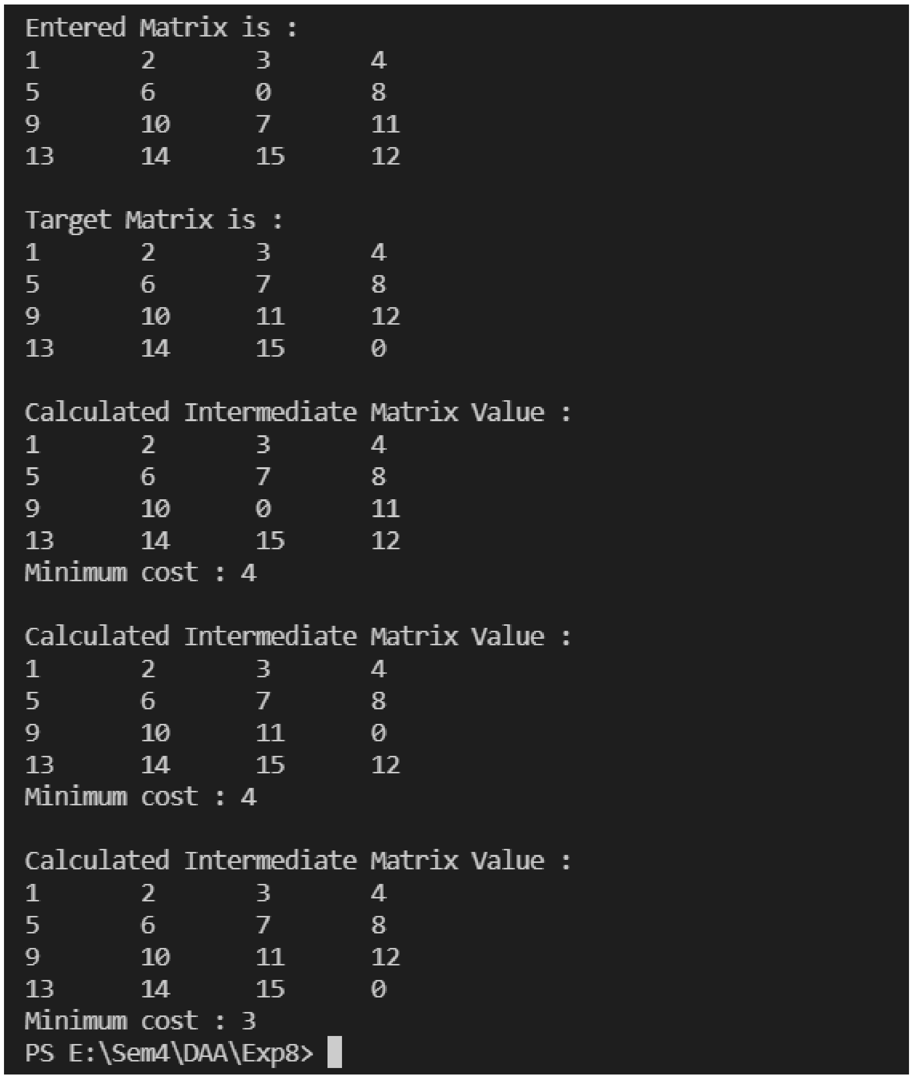
RESULT (SNAPSHOT)



```
Enter the matrix to be solved,space with zero :
1 2 3 4
5 6 0 8
9 10 7 11
13 14 15 12

Enter the target matrix,space with zero :

1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 0
```



```
Entered Matrix is :
1       2       3       4
5       6       0       8
9       10      7       11
13      14      15      12

Target Matrix is :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      0       11
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      0
13      14      15      12
Minimum cost : 4

Calculated Intermediate Matrix Value :
1       2       3       4
5       6       7       8
9       10      11      12
13      14      15      0
Minimum cost : 3
PS E:\Sem4\DAA\Exp8>
```

CONCLUSION:	Through this experiment I understood the concept of Branch And Bound and how to use it to solve 15 puzzle problem using Least Cost function to select the node with least cost.