NAME:	Hardik Garg
UID:	2021300036
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT NO:	08
AIM:	To implement Branch and bound strategy
ALGORITHM:	**Branch and Bound Strategy Algorithm** Algorithm for implementing the 15 puzzle problem in C: Initialize variables m, n, a[10][10], t[10][10], temp[10][10], r[10][10], x, y, d, dmin, and l. Read the matrix to be solved from the user, and the target matrix. Print the entered matrix and the target matrix. While the matrix is not solved, do the following steps: a. Set d to a high value of 1000. b. Find the position of the blank space (0) in the matrix. c. Copy the current matrix to the temp matrix. d. Move the blank space upwards, and calculate the Manhattan distance between the new matrix and the target matrix. If the distance is less than the previous distance d, update the value of d and copy the new matrix to r. e. Move the blank space downwards, and calculate the Manhattan distance between the new matrix and the target matrix. If the distance is less than the previous distance d, update the value of d and copy the new matrix to r. f. Move the blank space to the right, and calculate the Manhattan distance between the new matrix and the target matrix. If the distance is less than the previous distance d, update the value of d and copy the new matrix to r.

	g. Move the blank space to the left, and calculate the Manhattan distance between the new matrix and the target matrix. If the distance is less than the previous distance d, update the value of d and copy the new matrix to r. h. Print the calculated intermediate matrix value. i. Copy the values from the r matrix to the current matrix and reset the temp matrix. j. Increment the value of l. When the matrix is solved, print the final matrix and exit the program.

CODE	**Source Code**
	
	#include<stdio.h>
	#include<conio.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;
                }
        }

//To move upwards
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];
}

//To move downwards
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];
}

//To move right side

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];
}

//To move left

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;
                        }

        }

        getch();
}
```

Output	

```
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

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

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

Conclusion:	Thus we have implemented branch and bound method and we have solved the 15 puzzle problem using branch and bound strategy