Name: Amal Thundiyil
Department: IT
Batch: D
UID: 2020400066

Aim: Implement 15 / 8 Puzzle Problem using LCBB . Show the state space tree generated along with the cost of each node generated.

Program:

```
import java.io.*;
import java.util.*;
import static java.lang.Math.*;

public class Puzzle15 {
        static Reader fs;
        static PrintWriter pw;
        static final int n = 4;

        static void solve() throws IOException{
                int in[][] = new int[n][n];
      for(int i = 0 ; i < n ; i++) {
         for(int j = 0 ; j < n ; j++) {
            in[i][j] = fs.nextInt();
         }
      }
      int fin[][] = new int[n][n];
      int cnt = 1;
      for(int i = 0 ; i < n ; i++) {
         for(int j = 0 ; j < n ; j++) {
            fin[i][j] = cnt++;
         }
      }
      fin[n-1][n-1]= 0;
      int x = -1, y = -1;
      for(int i = 0 ; i < n ; i++) {
         for(int j = 0 ; j < n ; j++) {
            if(in[i][j] == 0) {
               x = i;
               y = j;
               break;
            }
         }
      }
```

```
        findSolution(in, fin, 0, -1, x, y);
        print(in);
            }

    static boolean findSolution(int in[][], int fin[][], int depth, int restrict, int i, int j) {
        PriorityQueue<int[]> q = new PriorityQueue<>((p1, p2) -> Integer.compare(p1[0],
p2[0]));
        if(heuristicFunction(in, fin) == 0) {
            return true;
        }
        if(i - 1 >= 0 && restrict != 3) {
            swap(in, i, j, i-1, j);
            int tmp[] = new int[]{heuristicFunction(in, fin) + depth, i-1, j, 0};
            q.add(tmp);
            swap(in, i, j, i-1, j);
        }
        if(j - 1 >= 0 && restrict != 2) {
            swap(in, i, j, i, j-1);
            int tmp[] = new int[]{heuristicFunction(in, fin) + depth, i, j-1, 1};
            q.add(tmp);
            swap(in, i, j, i, j-1);
        }
        if(i + 1 < n && restrict != 1) {
            swap(in, i, j, i+1, j);
            int tmp[] = new int[]{heuristicFunction(in, fin) + depth, i+1, j, 2};
            q.add(tmp);
            swap(in, i, j, i+1, j);
        }
        if(j + 1 < n && restrict != 0) {
            swap(in, i, j, i, j+1);
            int tmp[] = new int[]{heuristicFunction(in, fin) + depth, i, j+1, 3};
            q.add(tmp);
            swap(in, i, j, i, j+1);
        }
        for(int [] tmp: q) {
            swap(in, i, j, tmp[1], tmp[2]);
            boolean ans = findSolution(in, fin, depth + 1, tmp[3], tmp[1], tmp[2]);
            if(ans) {
                swap(in, i, j, tmp[1], tmp[2]);
                pw.println("Cost:" + tmp[0]);
                print(in);
                return true;
            }
            swap(in, i, j, tmp[1], tmp[2]);
```

```
    }
    return false;
}

static void print(int in[][]) {
    for(int i = 0 ; i < n ; i++) {
        for(int j = 0 ; j < n; j++) {
            pw.print(in[i][j] + " ");
        }
        pw.println();
    }
    pw.println();
}

static void swap(int in[][], int i, int j, int x, int y){
    int temp = in[i][j];
    in[i][j] = in[x][y];
    in[x][y] = temp;
}

static int heuristicFunction(int in[][], int fin[][]) {
    int ans = 0;
    for(int i= 0 ; i < n ; i++) {
        for(int j = 0 ; j < n; j++) {
            ans += in[i][j] != fin[i][j] ? 1 : 0;
        }
    }
    return ans;
}

        public static void main(String args[]) throws Exception{
                System.setErr(new PrintStream("error.txt"));
    System.setIn(new FileInputStream("input.txt"));
                fs = new Reader();
                pw = new PrintWriter(System.out);
                solve();
                pw.close();
        }

        static class Reader {
                BufferedReader br;
                StringTokenizer st;
                Reader() {
                        br = new BufferedReader(new InputStreamReader(System.in));
```

```
            st = new StringTokenizer("");
        }

        void fill() throws IOException{
            st = new StringTokenizer(br.readLine());
        }

        void check() throws IOException{
            if(!st.hasMoreTokens()) fill();
        }

        int nextInt() throws IOException{
            check();
            return Integer.parseInt(st.nextToken());
        }

        double nextDouble() throws IOException{
            check();
            return Double.parseDouble(st.nextToken());
        }

        long nextLong() throws IOException {
            check();
            return Long.parseLong(st.nextToken());
        }

        int [] readArray(int n) throws IOException{
            int a[] = new int[n];
            for(int i = 0 ; i < n ;i++) a[i] = nextInt();
            return a;
        }

        String next() throws IOException {
            check();
            return st.nextToken();
        }
    }
}
```

Output:

```
amal@ubuntu ~/Documents/Labs/DAA_LAB main java Puzzle15
Cost:3
1 2 3 4
5 6 7 8
9 10 11 0
13 14 15 12

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

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

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

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