AI LAB ASSIGNMENT 6

NAME: PRATHAPANI SATWIKA

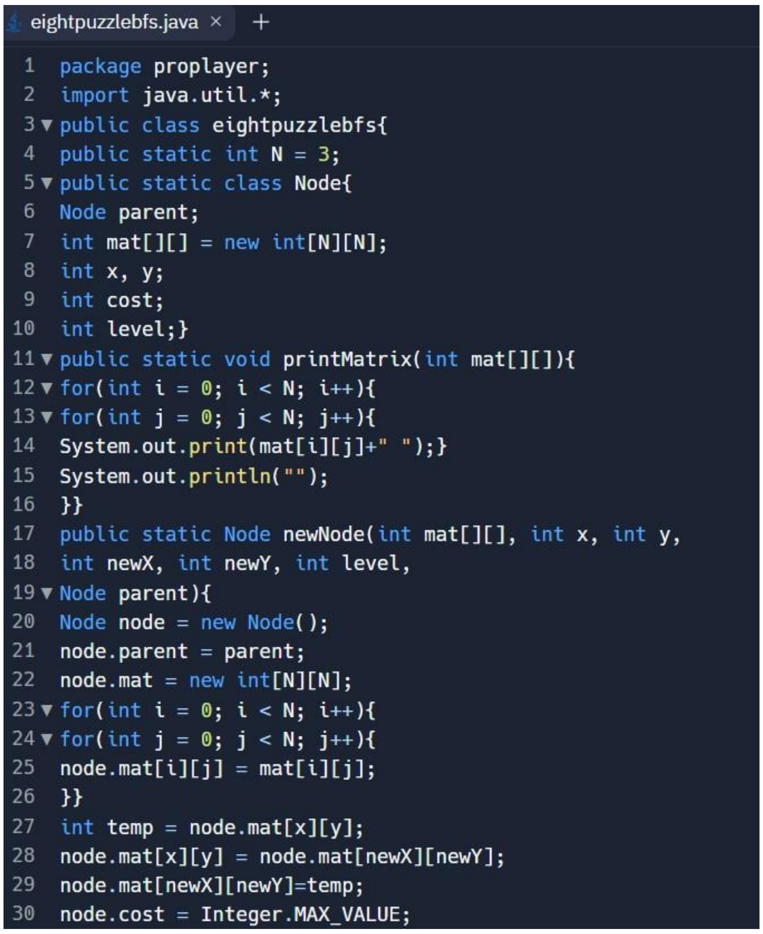
REG NO: 20BCD7160

Solve 8-puzzle problem by using best first search

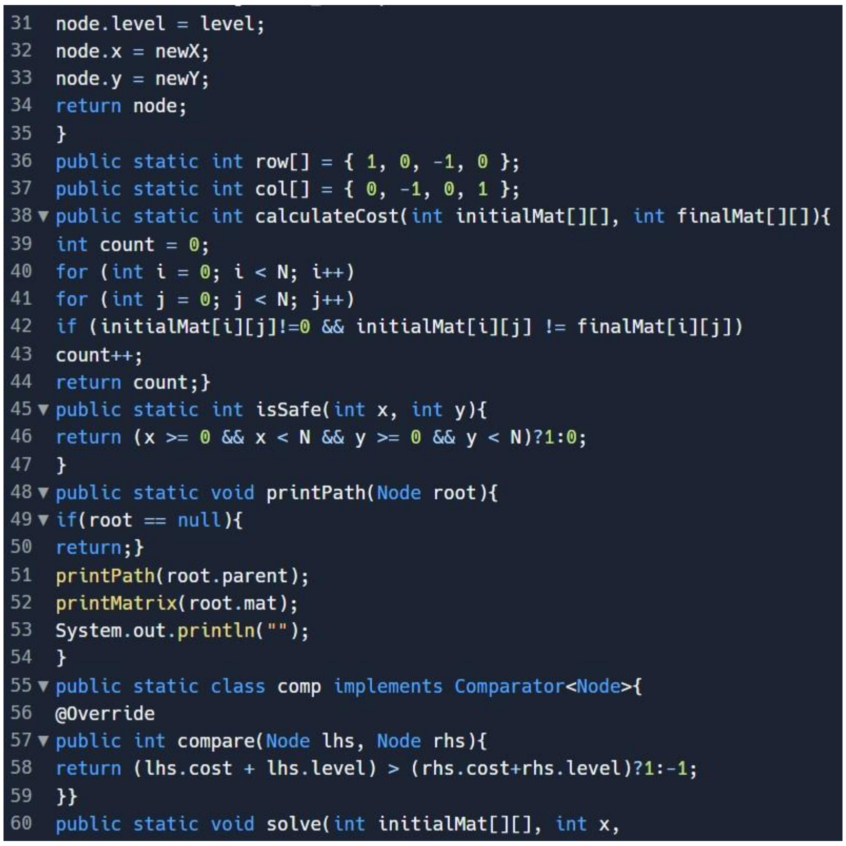
CODE:

```
package proplayer; import
java.util.*; public class
eightpuzzlebfs{ public
static int N = 3; public
static class Node{ Node
parent; int mat[][] = new
int[N][N]; int x, y; int cost;
int level;}
public static void printMatrix(int mat[][]){
for(int i = 0; i < N; i++){ for(int j = 0; j < N;
j++){
System.out.print(mat[i][j]+" ");}
System.out.println("");
}}
public static Node newNode(int mat[][], int x, int y,
int newX, int newY, int level,
Node parent){
Node node = new Node();
node.parent = parent;
node.mat = new int[N][N];
for(int i = 0; i < N; i++){ for(int
j = 0; j < N; j++){
node.mat[i][j] = mat[i][j];
}}
int temp = node.mat[x][y];
node.mat[x][y] = node.mat[newX][newY];
node.mat[newX][newY]=temp; node.cost
= Integer.MAX_VALUE; node.level = level;
node.x = newX; node.y = newY; return
node;
}
public static int row[] = { 1, 0, -1, 0 }; public
static int col[] = { 0, -1, 0, 1 };
public static int calculateCost(int initialMat[][], int finalMat[][]){
int count = 0; for (int i = 0; i < N; i++) for (int j = 0; j < N; j++)
if (initialMat[i][j]!=0 && initialMat[i][j] != finalMat[i][j])
count++; return count;}
public static int isSafe(int x, int y){
return (x >= 0 && x < N && y >= 0 && y < N)?1:0;
}
public static void printPath(Node root){
if(root == null){ return;}
printPath(root.parent);
printMatrix(root.mat);
System.out.println("");
```

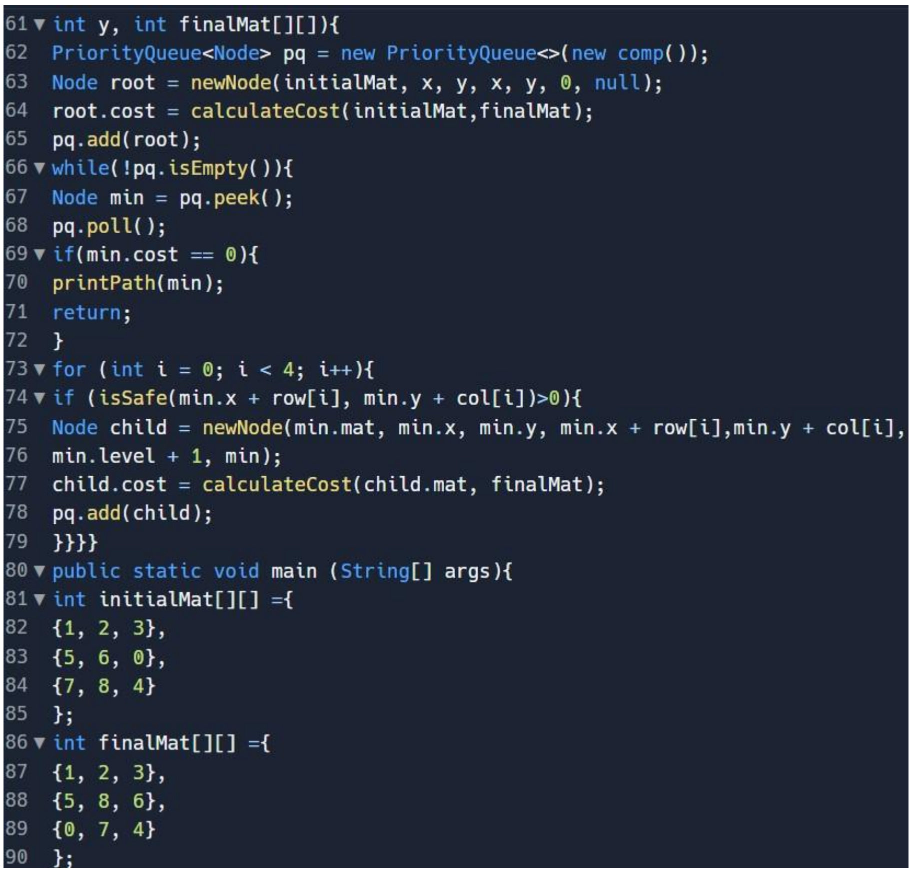
```
}
public static class comp implements Comparator<Node>{
@Override
public int compare(Node lhs, Node rhs){ return
(lhs.cost + lhs.level) > (rhs.cost+rhs.level)?1:-1;
}}
public static void solve(int initialMat[][], int x, int
y, int finalMat[][]){
PriorityQueue<Node> pq = new PriorityQueue<>(new comp());
Node root = newNode(initialMat, x, y, x, y, 0, null); root.cost =
calculateCost(initialMat,finalMat); pq.add(root);
while(!pq.isEmpty()){ Node min = pq.peek(); pq.poll();
if(min.cost == 0){ printPath(min); return;

}
for (int i = 0; i < 4; i++){
if (isSafe(min.x + row[i], min.y + col[i])>0){
Node child = newNode(min.mat, min.x, min.y, min.x + row[i],min.y + col[i],
min.level + 1, min);
child.cost = calculateCost(child.mat, finalMat);
pq.add(child);
}}}}
public static void main (String[] args){ int
initialMat[][] ={
{1, 2, 3},
{5, 6, 0},
{7, 8, 4}
};
int finalMat[][] ={
{1, 2, 3},
{5, 8, 6},
{0, 7, 4}
};
int x = 1, y = 2;
solve(initialMat, x, y,finalMat);
}
}
```


eightpuzzlebfs.java × +

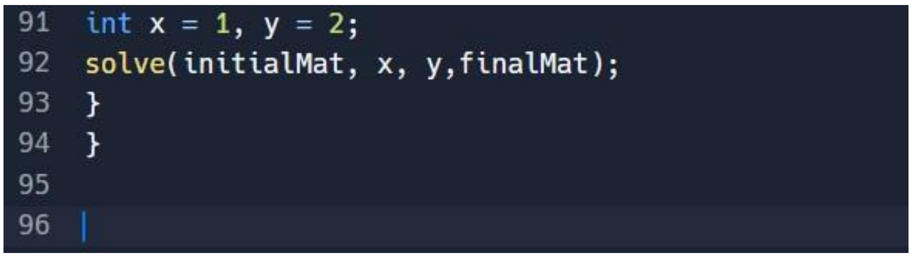
```java
package proplayer;
import java.util.*;
public class eightpuzzlebfs{
public static int N = 3;
public static class Node{
Node parent;
int mat[][] = new int[N][N];
int x, y;
int cost;
int level;}
public static void printMatrix(int mat[][]){
for(int i = 0; i < N; i++){
for(int j = 0; j < N; j++){
System.out.print(mat[i][j]+" ");}
System.out.println("");
}}
public static Node newNode(int mat[][], int x, int y,
int newX, int newY, int level,
Node parent){
Node node = new Node();
node.parent = parent;
node.mat = new int[N][N];
for(int i = 0; i < N; i++){
for(int j = 0; j < N; j++){
node.mat[i][j] = mat[i][j];
}}
int temp = node.mat[x][y];
node.mat[x][y] = node.mat[newX][newY];
node.mat[newX][newY]=temp;
node.cost = Integer.MAX_VALUE;
```



```java
node.level = level;
node.x = newX;
node.y = newY;
return node;
}
public static int row[] = { 1, 0, -1, 0 };
public static int col[] = { 0, -1, 0, 1 };
public static int calculateCost(int initialMat[][], int finalMat[][]){
int count = 0;
for (int i = 0; i < N; i++)
for (int j = 0; j < N; j++)
if (initialMat[i][j]!=0 && initialMat[i][j] != finalMat[i][j])
count++;
return count;}
public static int isSafe(int x, int y){
return (x >= 0 && x < N && y >= 0 && y < N)?1:0;
}
public static void printPath(Node root){
if(root == null){
return;}
printPath(root.parent);
printMatrix(root.mat);
System.out.println("");
}
public static class comp implements Comparator<Node>{
@Override
public int compare(Node lhs, Node rhs){
return (lhs.cost + lhs.level) > (rhs.cost+rhs.level)?1:-1;
}}
public static void solve(int initialMat[][], int x,
```

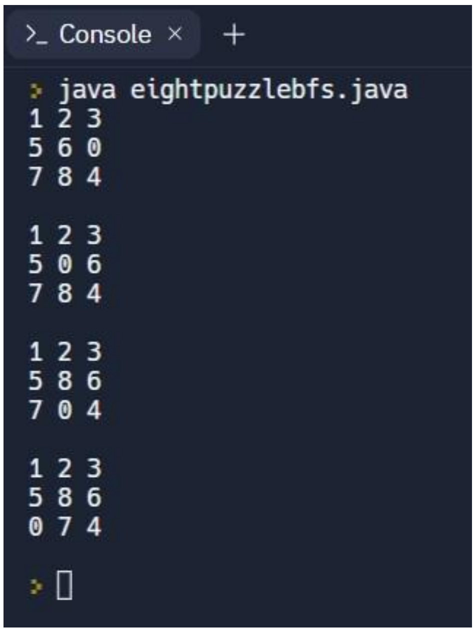


```java
int y, int finalMat[][]){
PriorityQueue<Node> pq = new PriorityQueue<>(new comp());
Node root = newNode(initialMat, x, y, x, y, 0, null);
root.cost = calculateCost(initialMat,finalMat);
pq.add(root);
while(!pq.isEmpty()){
Node min = pq.peek();
pq.poll();
if(min.cost == 0){
printPath(min);
return;
}
for (int i = 0; i < 4; i++){
if (isSafe(min.x + row[i], min.y + col[i])>0){
Node child = newNode(min.mat, min.x, min.y, min.x + row[i],min.y + col[i],
min.level + 1, min);
child.cost = calculateCost(child.mat, finalMat);
pq.add(child);
}}}}
public static void main (String[] args){
int initialMat[][] ={
{1, 2, 3},
{5, 6, 0},
{7, 8, 4}
};
int finalMat[][] ={
{1, 2, 3},
{5, 8, 6},
{0, 7, 4}
};
```



```java
int x = 1, y = 2;
solve(initialMat, x, y,finalMat);
}
}
```

OUTPUT:



```
> java eightpuzzlebfs.java
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
5 8 6
7 0 4

1 2 3
5 8 6
0 7 4

> 
```