import java.util.\*;

class astar {

public static int N = 3;

public static class Node {

Node parent;

int[][] m = new int[N][N];

int x, y;

int cost;

int level;

}

public static void printMatrix(int[][] m) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

System.out.print(m[i][j] + " ");

}

System.out.println("");

}

}

public static Node newNode(int[][] m, int x, int y, int newX, int newY, int level, Node parent) {

Node node = new Node();

node.parent = parent;

node.m = new int[N][N];

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

node.m[i][j] = m[i][j];

}

}

int temp = node.m[x][y];

node.m[x][y] = node.m[newX][newY];

node.m[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[][] initial, int[][] goal) {

int count = 0;

for (int i = 0; i < N; i++)

for (int j = 0; j < N; j++)

if (initial[i][j] != 0 && initial[i][j] != goal[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.m);

System.out.println("");

}

public static class comp implements Comparator<Node> {

public int compare(Node lhs, Node rhs) {

return (lhs.cost + lhs.level) > (rhs.cost + rhs.level) ? 1 : -1;

}

}

public static void solve(int[][] initial, int x, int y, int[][] goal) {

PriorityQueue<Node> pq = new PriorityQueue<>(new comp());

Node root = newNode(initial, x, y, x, y, 0, null);

root.cost = calculateCost(initial, goal);

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.m, min.x, min.y, min.x + row[i], min.y + col[i], min.level + 1, min);

child.cost = calculateCost(child.m, goal);

pq.add(child);

}

}

}

}

public static void main(String args[]) {

Scanner scanner = new Scanner(System.in);

int[][] initial = new int[N][N];

int[][] goal = new int[N][N];

int x = -1, y = -1; // Initialize x and y

System.out.println("Enter the initial state matrix:");

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

initial[i][j] = scanner.nextInt();

if (initial[i][j] == 0) {

x = i;

y = j;

}

}

}

System.out.println("Enter the goal state matrix:");

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

goal[i][j] = scanner.nextInt();

}

}

solve(initial, x, y, goal);

}

}