**Code:**

#include <bits/stdc++.h>

using namespace std;

#define N 3

struct Node

{

Node\* parent;

int mat[N][N];

int x, y;

int cost;

int level;

};

void printMatrix(int mat[N][N])

{

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

{

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

printf("%d ", mat[i][j]);

printf("\n");

}

}

Node\* newNode(int mat[N][N], int x, int y, int newX,

int newY, int level, Node\* parent)

{

Node\* node = new Node;

node->parent = parent;

memcpy(node->mat, mat, sizeof node->mat);

swap(node->mat[x][y], node->mat[newX][newY]);

node->cost = INT\_MAX;

node->level = level;

node->x = newX;

node->y = newY;

return node;

}

// bottom, left, top, right

int row[] = { 1, 0, -1, 0 };

int col[] = { 0, -1, 0, 1 };

int calculateCost(int initial[N][N], int final[N][N])

{

int count = 0;

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

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

if (initial[i][j] && initial[i][j] != final[i][j])

count++;

return count;

}

int isSafe(int x, int y)

{

return (x >= 0 && x < N && y >= 0 && y < N);

}

void printPath(Node\* root)

{

if (root == NULL)

return;

printPath(root->parent);

printMatrix(root->mat);

printf("\n");

}

struct comp

{

bool operator()(const Node\* lhs, const Node\* rhs) const

{

return (lhs->cost + lhs->level) > (rhs->cost + rhs->level);

}

};

void solve(int initial[N][N], int x, int y,

int final[N][N])

{

priority\_queue<Node\*, std::vector<Node\*>, comp> pq;

Node\* root = newNode(initial, x, y, x, y, 0, NULL);

root->cost = calculateCost(initial, final);

pq.push(root);

while (!pq.empty())

{

Node\* min = pq.top();

pq.pop();

if (min->cost == 0)

{

printPath(min);

return;

}

for (int i = 0; i < 4; i++)

{

if (isSafe(min->x + row[i], min->y + col[i]))

{

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, final);

pq.push(child);

}

}

}

}

int main()

{

int initial[N][N] =

{

{1, 2, 3},

{5, 6, 0},

{7, 8, 4}

};

int final[N][N] =

{

{1, 2, 3},

{4, 5, 6},

{7, 8, 0}

};

int x = 1, y = 2;

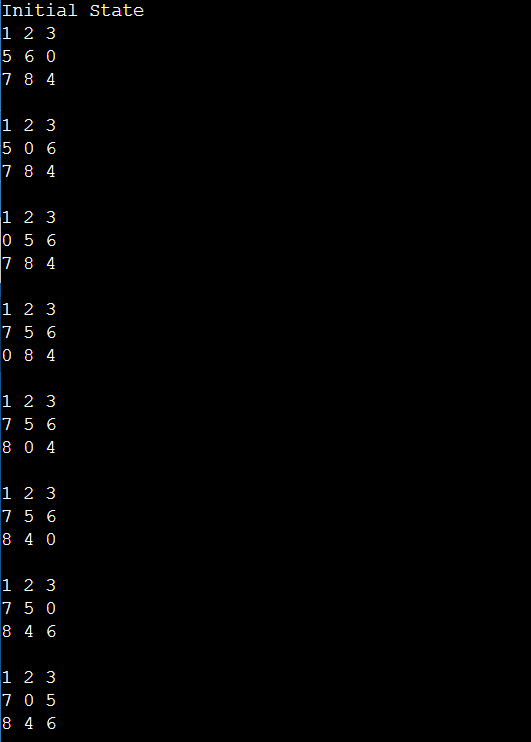
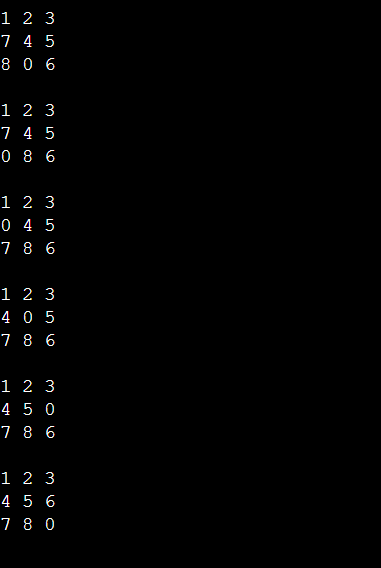
printf("Initial State\n");

solve(initial, x, y, final);

return 0;

}

**Output:**

** **