Assignment No 2

A* Algorithm:

```
#include <bits/stdc++.h>
#define n 3
const bool SUCCESS = true;
using namespace std;
using i64 = long long int;

class state {
public:
    int board[n][n], g, f;
    state* came_from;
    state () {
        g = 0;
        f = 0;
        came_from = NULL;
    }
    static int heuristic (state from, state to) {
        int ret = 0;
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (from.board[i][j] != to.board[i][j])
                    ret++;
        return ret;
    }
    bool operator == (state a) {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++)
                if (this->board[i][j] != a.board[i][j])
                    return false;
        return true;
    }
    void print () {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++)
                cout << board[i][j] << " ";
            cout << endl;
        }
        cout << "g = " << g << " | f = " << f << endl;
    }
};

vector <state> output;
bool lowerF (state a, state b) {
    return a.f < b.f;
}

bool isinset (state a, vector <state> b) {
    for (int i = 0; i < b.size(); i++)
        if (a == b[i])
```

```
            return true;
    return false;
}

void addNeighbor (state current, state goal, int newi, int newj, int posi, int posj, vector <state>&
openset, vector <state> closedset) {
    state newstate = current;
    swap (newstate.board[newi][newj], newstate.board[posi][posj]);
    if (!isinset(newstate, closedset) && !isinset(newstate, openset)) {
        newstate.g = current.g + 1;
        newstate.f = newstate.g + state :: heuristic(newstate, goal);
        state* temp = new state();
        *temp = current;
        newstate.came_from = temp;
        openset.push_back(newstate);
    }
}

void neighbors (state current, state goal, vector <state>& openset, vector <state>& closedset) {
    int i, j, posi , posj;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (current.board[i][j] == 0) {
                posi = i;
                posj = j;
                break;
            }
    i = posi, j = posj;
    if (i - 1 >= 0)
        addNeighbor(current, goal, i - 1, j, posi, posj, openset, closedset);
    if (i + 1 < n)
        addNeighbor(current, goal, i + 1, j, posi, posj, openset, closedset);
    if (j + 1 < n)
        addNeighbor(current, goal, i, j + 1, posi, posj, openset, closedset);
    if (j - 1 >= 0)
        addNeighbor(current, goal, i, j - 1, posi, posj, openset, closedset);
}

bool reconstruct_path(state current, vector<state> &came_from) {
    state *temp = &current;
    while (temp != NULL) {
        came_from.push_back(*temp);
        temp = temp->came_from;
    }
    return SUCCESS;
}

bool astar (state start, state goal) {
    vector <state> openset;
    vector <state> closedset;
    state current;
    start.g = 0;
    start.f = start.g + state :: heuristic(start, goal);
```

```
    openset.push_back(start);
    while (!openset.empty()) {
        sort(openset.begin(), openset.end(), lowerF);
        current = openset[0];
        if (current == goal)
            return reconstruct_path(current, output);
        openset.erase(openset.begin());
        closedset.push_back(current);
        neighbors(current, goal, openset, closedset);
    }
    return !SUCCESS;
}

int main () {
    state start, goal;
//   freopen("in.txt", "r", stdin);
    cout<<"Enter Start state: "<<endl;
    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) cin >> start.board[i][j];
    cout<<"Enter Goal state: "<<endl;
    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) cin >> goal.board[i][j];
    cout<<"-------ANSWER----------"<<endl;
    if (astar(start, goal) == SUCCESS) {
        for (int i = output.size() - 1; i >= 0; i--)
            output[i].print();
        cout << "----------Success----------" << endl;
    }
    else cout << "------FAIL-----" << endl;
    return 0;
}
```

```
PS E:\LP2> cd "e:\LP2\" ; if ($?) { g++ temp.cpp -o temp } ; if ($?) { .\temp }
Enter Start state:
2 8 3
1 6 4
7 0 5
Enter Goal state:
1 2 3
8 0 4
7 6 5
-------ANSWER----------
2 8 3
1 6 4
7 0 5
g = 0 | f = 5
2 8 3
1 0 4
7 6 5
g = 1 | f = 4
2 0 3
1 8 4
7 6 5
g = 2 | f = 6
0 2 3
1 8 4
7 6 5
g = 3 | f = 6
1 2 3
0 8 4
7 6 5
g = 4 | f = 6
1 2 3
8 0 4
7 6 5
g = 5 | f = 5
----------Success----------
PS E:\LP2>
```