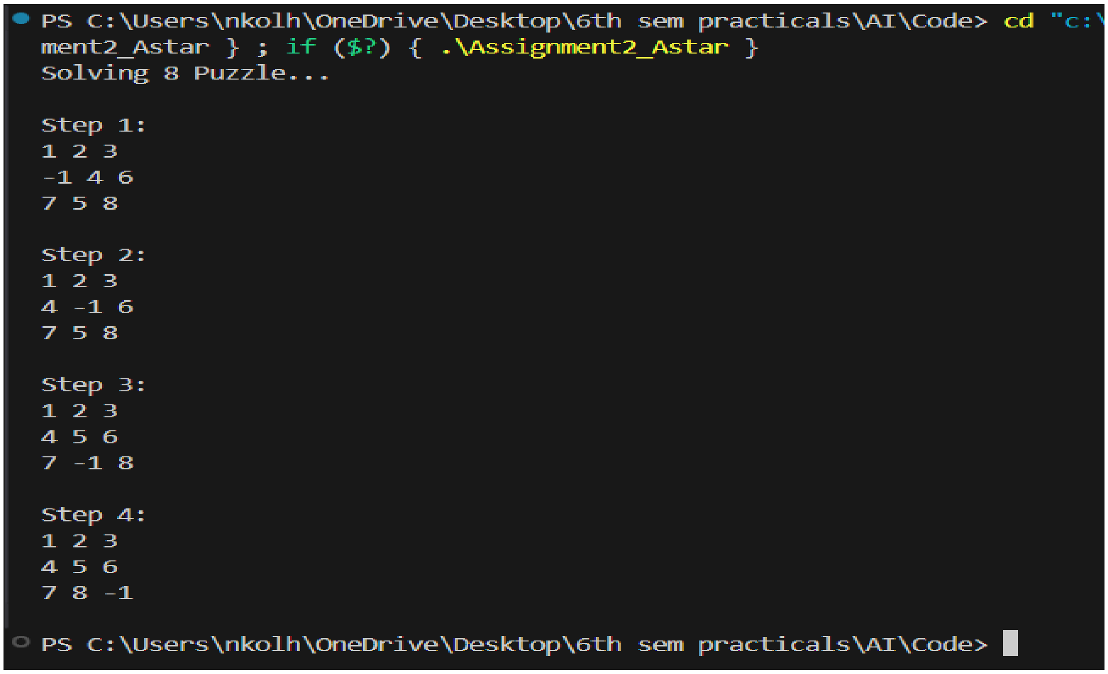
Assignment No 2

Code

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

const int N = 3;

// Function to count misplaced tiles (heuristic)
int gn(vector<vector<int>> state, vector<vector<int>> goal) {
    int count = 0;
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            if (state[i][j] != -1 && state[i][j] != goal[i][j])
                count++;
    return count;
}

// Find position of blank tile (-1)
pair<int, int> findBlank(vector<vector<int>>& state) {
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            if (state[i][j] == -1)
                return {i, j};
    return {-1, -1};
}

// Move functions
vector<vector<int>> moveLeft(vector<vector<int>> state, pair<int, int> pos) {
    if (pos.second == 0) return {};
    swap(state[pos.first][pos.second], state[pos.first][pos.second - 1]);
    return state;
}

vector<vector<int>> moveRight(vector<vector<int>> state, pair<int, int> pos) {
    if (pos.second == N - 1) return {};
    swap(state[pos.first][pos.second], state[pos.first][pos.second + 1]);
    return state;
}

vector<vector<int>> moveUp(vector<vector<int>> state, pair<int, int> pos) {
    if (pos.first == 0) return {};
    swap(state[pos.first][pos.second], state[pos.first - 1][pos.second]);
    return state;
}

vector<vector<int>> moveDown(vector<vector<int>> state, pair<int, int> pos) {
    if (pos.first == N - 1) return {};
    swap(state[pos.first][pos.second], state[pos.first + 1][pos.second]);
    return state;
}

// Print all steps
void printMatrix(const vector<vector<vector<int>>>& explored) {
    cout << endl;
    int counter = 1;
    for (const auto& matrix : explored) {
        cout << "Step " << counter++ << ":\n";
        for (const auto& row : matrix) {
            for (int val : row)
                cout << val << " ";
            cout << endl;
        }
        cout << endl;
    }
```

```
}

// Main solving function
void eightPuzzle(vector<vector<int>> initial, vector<vector<int>> goal) {
    int hn = 0;
    vector<vector<vector<int>>> explored;

    while (true) {
        explored.push_back(initial);
        if (initial == goal) break;

        hn++;
        pair<int, int> pos = findBlank(initial);

        vector<vector<int>> left = moveLeft(initial, pos);
        vector<vector<int>> right = moveRight(initial, pos);
        vector<vector<int>> up = moveUp(initial, pos);
        vector<vector<int>> down = moveDown(initial, pos);

        int fnl = 1e9, fnr = 1e9, fnu = 1e9, fnd = 1e9;

        if (!left.empty()) fnl = hn + gn(left, goal);
        if (!right.empty()) fnr = hn + gn(right, goal);
        if (!up.empty()) fnu = hn + gn(up, goal);
        if (!down.empty()) fnd = hn + gn(down, goal);

        int minfn = min({fnl, fnr, fnu, fnd});

        if (fnl == minfn && find(explored.begin(), explored.end(), left) == explored.end()) {
            initial = left;
        } else if (fnr == minfn && find(explored.begin(), explored.end(), right) == explored.end()) {
            initial = right;
        } else if (fnu == minfn && find(explored.begin(), explored.end(), up) == explored.end()) {
            initial = up;
        } else if (fnd == minfn && find(explored.begin(), explored.end(), down) == explored.end()) {
            initial = down;
        }
    }

    printMatrix(explored);
}

int main() {


    vector<vector<int>> start = {{1, 2, 3},{-1, 4, 6},{7, 5, 8}};
    vector<vector<int>> goal = {{1, 2, 3},{4, 5, 6},{7, 8, -1}};

    cout << "Solving 8 Puzzle...\n";
    eightPuzzle(start, goal);



    return 0;
}
```

Output :-



```
PS C:\Users\nkolh\OneDrive\Desktop\6th sem practicals\AI\Code> cd "c:\
ment2_Astar } ; if ($?) { .\Assignment2_Astar }
Solving 8 Puzzle...

Step 1:
1 2 3
-1 4 6
7 5 8

Step 2:
1 2 3
4 -1 6
7 5 8

Step 3:
1 2 3
4 5 6
7 -1 8

Step 4:
1 2 3
4 5 6
7 8 -1

PS C:\Users\nkolh\OneDrive\Desktop\6th sem practicals\AI\Code>
```