Name: Nitin Choudhary
Roll no: 50
Class: TY-IT A B3

8-Puzzle using Steepest Hill Climbing:

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

using namespace std;

int misplaced_tiles(const vector<vector<int>> &state, const
vector<vector<int>> &goal_state)
{
    int misplaced_count = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (state[i][j] != goal_state[i][j])
            {
                misplaced_count++;
            }
        }
    }
    return misplaced_count;
}

vector<vector<vector<int>>>
get_next_states(const vector<vector<int>> &state)
{
    int moves[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
    int zero_row = -1, zero_col = -1;

    for (int i = 0; i < 3 && zero_row == -1; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            if (state[i][j] == 0)
            {
                zero_row = i;
                zero_col = j;
                break;
```

```
            }
        }
    }

    vector<vector<vector<int>>>
    next_states;

    for (int i = 0; i < 4; i++)
    {
        int new_row = zero_row + moves[i][0];
        int new_col = zero_col + moves[i][1];

        if (new_row >= 0 && new_row < 3 && new_col >= 0 && new_col < 3)
        {
            vector<vector<int>> new_state = state;
            swap(new_state[zero_row][zero_col],
new_state[new_row][new_col]);
            next_states.push_back(new_state);
        }
    }
    return next_states;
}

int calculate_f(const vector<vector<int>> &state, const vector<vector<int>>
&goal_state, int a)
{
    return a + misplaced_tiles(state, goal_state);
}

int main()
{
    const vector<vector<int>> initial_state = {
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 0}};

    const vector<vector<int>> goal_state = {
        {1, 2, 3},
        {4, 8, 5},
        {0, 7, 6}};

    int a = 0;
    int initial_f = calculate_f(initial_state, goal_state, a);

    vector<vector<vector<int>>>
```

```
        next_states = get_next_states(initial_state);

    vector<vector<int>> best_state;
    int best_f = initial_f;

    cout << "Initial State:" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << initial_state[i][j] << " ";
        }
        cout << endl;
    }
    cout << "f(x) = " << initial_f << endl;

    cout << "\nPossible Next Moves and their f(x) values:" << endl;
    for (size_t k = 0; k < next_states.size(); k++)
    {
        int f_value = calculate_f(next_states[k], goal_state, a);
        cout << "f(x) = " << f_value << endl;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                cout << next_states[k][i][j] << " ";
            }
            cout << endl;
        }
        cout << "----------" << endl;

        if (f_value < best_f)
        {
            best_f = f_value;
            best_state = next_states[k];
        }
    }

    if (!best_state.empty() && best_f < initial_f)
    {
        cout << "\nBetter Initial State:" << endl;
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
```

```
                cout << best_state[i][j] << " ";
            }
            cout << endl;
        }
        cout << "f(x) of Best Next State: " << best_f << endl;
    }
    else
    {
        cout << "\nNo better state found." << endl;
    }

    return 0;
}
```

Output:

Initial State:
1 2 3
4 5 6
7 8 0
f(x) = 5

Possible Next Moves and their f(x) values:
f(x) = 5
1 2 3
4 5 6
7 0 8
---
f(x) = 4
1 2 3
4 5 0
7 8 6

Better Initial State:
1 2 3
4 5 0
7 8 6
f(x) of Best Next State: 4