VISHWAKARMA INSTITUTE OF TECHNOLOGY
COMPUTER ENGINEERING

Name: Abhinav Mahajan

Division: TY-C

Roll No: 15

Subject: Artificial Intelligence (AI)

LAB ASSIGNMENT NO – 3

Implementation of **Informed Strategies** for **8-Puzzle Game**.

The 8-puzzle problem is a sliding puzzle that involves arranging 8 numbered tiles in a 3x3 grid with one empty space, which aims to reach a specific goal configuration through tile swaps.

A* Algorithm:

Approach:

1. Set up initial and goal states as 2D vectors.
2. Initialize a priority queue containing pairs of current state and its path.
3. The priority queue (pq) holds pairs where the first element is the sum of "g" (depth) and "h" (heuristic) scores, and the second element is the current state of the puzzle.
4. Begin a loop that continues until the priority queue is empty.
5. Pop the current state and its path from the priority queue.
6. Iterate over the four possible neighbour positions around the empty cell: up, down, left, and right.
7. If current state matches goal state, print the path and stop.
8. Backtrack by swapping the empty cell back with its original position to restore the state for further exploration.
9. If the goal is found, the path leading to the goal state is printed.

Code:

```
#include <bits/stdc++.h>
using namespace std;

void findZero(vector<vector<int>> board, int &x, int &y){
    for (int i = 0; i < board.size(); i++){
        for (int j = 0; j < board.size(); j++){
            if (board[i][j] == 0){
                x = i;
                y = j;
                return;
            }
        }
    }
}

void printBoard(vector<vector<int>> board){
    for (int i = 0; i < board.size(); i++){
        for (int j = 0; j < board.size(); j++){
            cout << board[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

bool isGoalState(vector<vector<int>> &board, vector<vector<int>>
&goal) {
    return board == goal;
}

int findMisplacedTiles(vector<vector<int>> &board,
vector<vector<int>> &goal){
    int count = 0;
    for (int i = 0; i < board.size(); i++){
        for (int j = 0; j < board.size(); j++){
            if (board[i][j] != goal[i][j])
                count++;
        }
    }
    return count;
```

```
}

void aStar(vector<vector<int>> &board, vector<vector<int>> &goal,
int depth, int x, int y){
    priority_queue<pair<int, vector<vector<int>>>,
vector<pair<int, vector<vector<int>>>>, greater<pair<int,
vector<vector<int>>>>> pq;
    int g = depth;
    int h = findMisplacedTiles(board, goal);
    pq.push({(g + h), board});
    while (!pq.empty()){
        vector<vector<int>> curr = pq.top().second;
        pq.pop();
        printBoard(curr);
        int x, y;
        findZero(curr, x, y);
        if (isGoalState(curr, goal)){
            cout << "Goal State Reached" << endl;
            return;
        }

        int dx[] = {0, 0, -1, 1};
        int dy[] = {1, -1, 0, 0};

        for (int i = 0; i < 4; i++){
            int newX = x + dx[i];
            int newY = y + dy[i];
            if (newX >= 0 && newX < curr.size() && newY >= 0 &&
newY < curr.size()){
                swap(curr[x][y], curr[newX][newY]);
                g = depth + 1;
                h = findMisplacedTiles(curr, goal);
                pq.push({(g + h), curr});
                swap(curr[x][y], curr[newX][newY]);
            }
        }

        // ans.pop_back();
    }
    return;
}
```

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

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

    int x, y;
    findZero(initial, x, y);
    // vector<vector<vector<int>>> ans;
    aStar(initial, goal, 0, x, y);

    return 0;
}
```

Output:

```
PS C:\Users\mahaj> cd "d:\TY\AI\8 Puzzle\" ; if ($?) { g++ 8Puzzle_AStar.cpp -o 8Puzzle_AStar } ; if ($?) { .\8Puzzle_AStar }
 2 8 3
 1 6 4
 7 0 5

 2 8 3
 1 0 4
 7 6 5

 2 0 3
 1 8 4
 7 6 5

 0 2 3
 1 8 4
 7 6 5

 1 2 3
 0 8 4
 7 6 5

 1 2 3
 8 0 4
 7 6 5

 Goal State Reached
PS D:\TY\AI\8 Puzzle>
```