ARTIFICIAL INTELLIGENCE
8-PUZZLE PROBLEM

1 Introduction

The 8-puzzle problem is a well-known problem in artificial intelligence and operations research. It consists of a 3x3 grid containing eight numbered tiles and one blank tile. The objective is to arrange the tiles in a specific order by sliding them into the blank space.

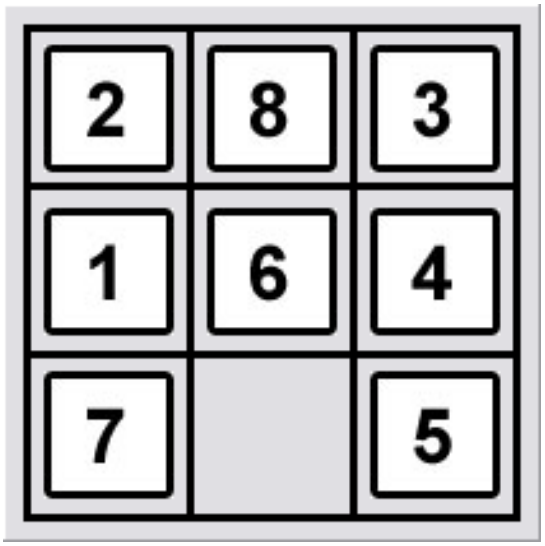

Figure 1: 8-puzzle problem

2 Aim

The aim of this assignment is to develop a C program that allows users to play the 8-puzzle game. The program will enable users to input an initial configuration of the puzzle and make moves to slide the tiles until they reach the goal state. The program will also count and display the number of moves taken to solve the puzzle.

3 Algorithm

The program follows these steps to implement the 8-puzzle game:

1. Initialize the game board and prompt the user for the initial configuration.
2. Validate the board configuration to ensure there is exactly one blank tile (represented by 0).
3. Enter the main game loop, which continues until the puzzle is solved:
   - Display the current state of the board.
   - Prompt the user for a move (up, down, left, or right).
   - Validate the user's input and ensure the move is within bounds.
   - Update the board by sliding the appropriate tile into the blank space.
   - Increment the move counter.
4. When the puzzle is solved, display the final board and the total number of moves.

4 Code Implementation

The following code implements the 8-puzzle game:

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 3

void printBoard(int board[SIZE][SIZE]);
int isSolved(int board[SIZE][SIZE]);
void makeMove(int board[SIZE][SIZE], int move);
int findZero(int board[SIZE][SIZE], int *zeroX, int *zeroY);
void swap(int *a, int *b);
void inputBoard(int board[SIZE][SIZE]);

int main() {
    int board[SIZE][SIZE];
    int move;
    int zeroX, zeroY;
    int steps = 0;

    printf("Welcome␣to␣the␣8-puzzle␣game!\n");
    printf("Please␣enter␣the␣initial␣configuration␣of␣the␣board␣(0␣
       for␣empty␣space):\n");
    inputBoard(board);

    int zeroCount = 0;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j] == 0) zeroCount++;
```

```
        }
    }
    if (zeroCount != 1) {
        printf("Invalid␣board␣configuration.␣It␣must␣contain␣exactly
           ␣one␣empty␣space␣(0).\n");
        return 1;
    }

    while (!isSolved(board)) {
        printBoard(board);
        int validInput = 0;

        while (!validInput) {
            printf("Enter␣move␣(1␣=␣Up,␣2␣=␣Down,␣3␣=␣Left,␣4␣=␣
               Right):␣");
            if (scanf("%d", &move) != 1) {
                printf("Invalid␣input.␣Please␣enter␣a␣number.\n");
                while (getchar() != '\n');
                continue;
            }
            if (move < 1 || move > 4) {
                printf("Invalid␣move.␣Please␣enter␣a␣number␣between␣
                   1␣and␣4.\n");
            } else {
                validInput = 1;
            }
        }

        if (findZero(board, &zeroX, &zeroY)) {
            makeMove(board, move);
            steps++;
        } else {
            printf("Error␣finding␣empty␣space.\n");
        }
    }

    printBoard(board);
    printf("Congratulations!␣You've␣solved␣the␣puzzle␣in␣%d␣steps!\n
       ", steps);
    return 0;
}

void printBoard(int board[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (board[i][j] == 0) {
                printf("␣␣");
```

```
69              } else {
70                  printf("%2d␣", board[i][j]);
71              }
72          }
73          printf("\n");
74      }
75      printf("\n");
76  }
77
78  int isSolved(int board[SIZE][SIZE]) {
79      int correct[SIZE][SIZE] = {
80          {1, 2, 3},
81          {4, 5, 6},
82          {7, 8, 0}
83      };
84      for (int i = 0; i < SIZE; i++) {
85          for (int j = 0; j < SIZE; j++) {
86              if (board[i][j] != correct[i][j]) {
87                  return 0;
88              }
89          }
90      }
91      return 1;
92  }
93
94  void makeMove(int board[SIZE][SIZE], int move) {
95      int zeroX, zeroY;
96      if (!findZero(board, &zeroX, &zeroY)) {
97          printf("Error␣finding␣empty␣space.\n");
98          return;
99      }
100     int newX = zeroX, newY = zeroY;
101     switch (move) {
102         case 1:
103             newX--;
104             break;
105         case 2:
106             newX++;
107             break;
108         case 3:
109             newY--;
110             break;
111         case 4:
112             newY++;
113             break;
114     }
115     if (newX < 0 || newX >= SIZE || newY < 0 || newY >= SIZE) {
```

```
116         printf("Move␣out␣of␣bounds.\n");
117         return;
118     }
119     swap(&board[zeroX][zeroY], &board[newX][newY]);
120 }
121 
122 int findZero(int board[SIZE][SIZE], int *zeroX, int *zeroY) {
123     for (int i = 0; i < SIZE; i++) {
124         for (int j = 0; j < SIZE; j++) {
125             if (board[i][j] == 0) {
126                 *zeroX = i;
127                 *zeroY = j;
128                 return 1;
129             }
130         }
131     }
132     return 0;
133 }
134 
135 void swap(int *a, int *b) {
136     int temp = *a;
137     *a = *b;
138     *b = temp;
139 }
140 
141 void inputBoard(int board[SIZE][SIZE]) {
142     for (int i = 0; i < SIZE; i++) {
143         for (int j = 0; j < SIZE; j++) {
144             while (1) {
145                 printf("Enter␣value␣for␣cell␣(%d,␣%d):␣", i, j);
146                 if (scanf("%d", &board[i][j]) != 1) {
147                     printf("Invalid␣input.␣Please␣enter␣an␣integer.\
                        n");
148                     while (getchar() != '\n');
149                 } else if (board[i][j] < 0 || board[i][j] > 8) {
150                     printf("Invalid␣value.␣Enter␣a␣number␣between␣0␣
                        and␣8.\n");
151                 } else {
152                     break;
153                 }
154             }
155         }
156     }
157 }
```

5 Output

After entering a valid initial configuration for the 8-puzzle, the program will display the board after each move until the puzzle is solved. Here is an example output:

```
Welcome to the 8-puzzle game!
Please enter the initial configuration of the board (0 for empty space):
Enter value for cell (0, 0): 1
Enter value for cell (0, 1): 2
Enter value for cell (0, 2): 3
Enter value for cell (1, 0): 4
Enter value for cell (1, 1): 0
Enter value for cell (1, 2): 5
Enter value for cell (2, 0): 7
Enter value for cell (2, 1): 8
Enter value for cell (2, 2): 6
 1  2  3
 4     5
 7  8  6

Enter move (1 = Up, 2 = Down, 3 = Left, 4 = Right): 4
 1  2  3
 4  5
 7  8  6

Enter move (1 = Up, 2 = Down, 3 = Left, 4 = Right): 2
 1  2  3
 4  5  6
 7  8

Congratulations! You've solved the puzzle in 2 steps!
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} (
iot_a1@snucse-HP-280-Pro-G5-MT-Business-PC:~/Desktop/KALYANI_22011102017/ailab$
```

Figure 2: 8-puzzle problem output

6 Conclusion

The 8-puzzle problem is an engaging way to explore algorithms related to state-space search. This program provides an interactive way for users to engage with the problem and gain insights into the challenges of solving combinatorial puzzles.