ARTIFICIAL INTELLIGENCE

EXPERIMENT NO: 8

IMPLEMENTATION OF KNOWLEDGE REPRESENTATION SCHEMAS

Nikith Kumar Seemakurthi

RA1911003020480

AIM:

To implement knowledge representation schemes.

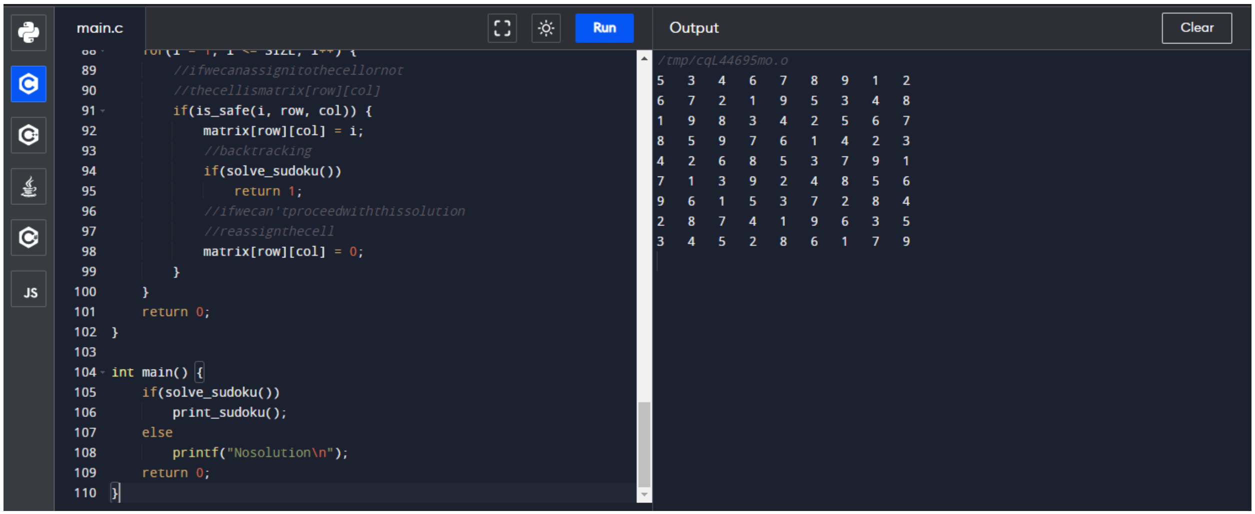
ALGORITHM:

1. Initialize a 9x9 array with numbers to represent the game of sudoku, where the numbers that are to be filled are represented with 0.
2. Define a variable called size with the value 9, this represents the size of array.
3. Create a function called is_safe that checks if a number can be placed in a corresponding row and column.
4. Create another function called number_unassigned, to check if there are any unsigned cells in row and column.
5. For every number from 1 to size, check if there are any unsigned cells and also check if the number is_safe to be placed in that cell using backtracking.
6. After looping through the numbers if there are noun signed cells then the puzzle is solved.
7. Print the solution to the puzzle.

SOURCE CODE:

```
#include <stdio.h>

#define SIZE 9

//sudokuproblem

int matrix[9][9]={

  {5,3,0,0,7,0,0,0,0},

  {6,0,0,1,9,5,0,0,0},

  {0,9,8,0,0,0,0,6,0},

  {8,0,0,0,6,0,0,0,3},

  {4,0,0,8,0,3,0,0,1},

  {7,0,0,0,2,0,0,0,6},

  {0,6,0,0,0,0,2,8,0},

  {0,0,0,4,1,9,0,0,5},
    {0,0,0,0,8,0,0,7,9}
};

//functiontoprintsudoku
void print_sudoku() {
    int i, j;
    for(i = 0; i < SIZE; i++) {
        for(j = 0; j < SIZE; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n\n");
    }
}

//functiontocheckifallcellsareassignedornot
//ifthereisanyunassignedcell
//thenthisfunctionwillchangethevaluesof
//rowandcolaccordingly
int number_unassigned(int* row, int* col) {
    int num_unassign = 0;
    int i, j;
    for(i = 0; i < SIZE; i++) {
        for(j = 0; j < SIZE; j++) {
            if(matrix[i][j] == 0) {
                //changingthevaluesofrowandcol
                *row = i;
                *col = j;
                //thereisoneormoreunassignedcells
                num_unassign = 1;
                return num_unassign;
            }
        }
    }
    return num_unassign;
}
//functiontocheckifwecanputa
//valueinapaticularcellornot
int is_safe(int n, int r, int c) {
    int i,j;
    //checkinginrow
    for(i = 0; i < SIZE; i++) {
        //thereisacellwithsamevalue
        if(matrix[r][i] == n)
            return 0;
    }
    //checkingcolumn
    for(i = 0; i < SIZE; i++) {
    //thereisacellwiththevalueequaltoi
        if(matrix[i][c] == n)
            return 0;
    }
    //checkingsubmatrix
    int row_start = (r/3)*3;
    int col_start = (c/3)*3;
    for(i = row_start; i < row_start+3; i++) {
        for(j = col_start; j < col_start+3; j++) {
            if(matrix[i][j] == n)
                return 0;
        }
    }
    return 1;
}

//functiontosolvesudoku
//usingbacktracking
int solve_sudoku() {
    int row;
    int col;
    //ifallcellsareassignedthenthesudokuisalreadysolved
    //passbyreferencebecausenumber_unassignedwillchange    thevaluesofrowandcol
    if(number_unassigned(&row, &col) == 0)
        return 1;
    int n, i;
    //numberbetween1to9
    for(i = 1; i <= SIZE; i++) {
        //ifwecanassignitothecellornot
        //thecellismatrix[row][col]
        if(is_safe(i, row, col)) {
            matrix[row][col] = i;
            //backtracking
            if(solve_sudoku())
                return 1;
            //ifwecan'tproceedwiththissolution
            //reassignthecell
            matrix[row][col] = 0;
        }
    }
    return 0;
}

int main() {
    if(solve_sudoku())
        print_sudoku();
    else
        printf("Nosolution\n");
    return 0;
}
```

OUTPUT:



```
    for(i = 1; i <= SIZE; i++) {
        //ifwecanassignitothecellornot
        //thecellismatrix[row][col]
        if(is_safe(i, row, col)) {
            matrix[row][col] = i;
            //backtracking
            if(solve_sudoku())
                return 1;
            //ifwecan'tproceedwiththissolution
            //reassignthecell
            matrix[row][col] = 0;
        }
    }
    return 0;
}

int main() {
    if(solve_sudoku())
        print_sudoku();
    else
        printf("Nosolution\n");
    return 0;
}
```

```
/tmp/cqL44695mo.o
5   3   4   6   7   8   9   1   2
6   7   2   1   9   5   3   4   8
1   9   8   3   4   2   5   6   7
8   5   9   7   6   1   4   2   3
4   2   6   8   5   3   7   9   1
7   1   3   9   2   4   8   5   6
9   6   1   5   3   7   2   8   4
2   8   7   4   1   9   6   3   5
3   4   5   2   8   6   1   7   9
```

RESULT:

Thus knowledge representation schemes have been implemented with Sudoku.