**CS-304-AI -LAB(LAB TASK-6)**

**ROLL-423135**

**NAME-HARSHITH SUDA**

**CODE-1**

#include<bits/stdc++.h>

using namespace std;

void printVectorOfVector(vector<vector<int>>&v){

    for(int i=0;i<v.size();i++){

        for(int j=0;j<v[0].size();j++){

*// cout<<"value of i:"<<i<<"value of j:"<<j<<" ";*

            cout<<v[i][j]<<" ";

        }

        cout<<endl;

    }

    return;

}

bool isValid(vector<vector<int>>&board,int i,int j,int z,vector<vector<bool>>&row,vector<vector<bool>>&col,vector<vector<bool>>&box){

    int s=(i/3)\*3+j/3;

    return !(row[i][z-1] || col[j][z-1] || box[s][z-1]);

}

bool sudoku(vector<vector<int>>&board,int i,int j,set<pair<int,int>>&s,vector<vector<bool>>&row,vector<vector<bool>>&col,vector<vector<bool>>&box){

    if(i==board.size()){

        return true;

    }

    if(j==board.size()){

        return sudoku(board,i+1,0,s,row,col,box);

    }

    if(s.find({i,j})!=s.end()){

        return sudoku(board,i,j+1,s,row,col,box);

    }

    for(int z=1;z<=board.size();z++){

        if(isValid(board,i,j,z,row,col,box)){

            board[i][j]=z-1;

            row[i][z-1]=1;

            col[j][z-1]=1;

            int seq=((i)/3)\*3+(j)/3;

            box[seq][z-1]=1;

            if(sudoku(board,i,j+1,s,row,col,box)){

                return true;

            }

            board[i][j]=0;

            row[i][z-1]=0;

            col[j][z-1]=0;

            box[seq][z-1]=0;

        }

    }

    return false;

}

void solveSudoku(vector<vector<string>>& board) {

*//intialize the set pairs and the bool rows,cols and box with the pre entered values in board*

    vector<vector<int>>integerBoard(board.size(),vector<int>(board.size()));

    vector<vector<bool>>row(9,vector<bool>(9,false));

    vector<vector<bool>>col(9,vector<bool>(9,false));

    vector<vector<bool>>box(9,vector<bool>(9,false));

    set<pair<int,int>>s;

    for(int i=0;i<board.size();i++){

        for(int j=0;j<board.size();j++){

            if(board[i][j]=="."){

                integerBoard[i][j]=0;

                continue;

            }

            char c=board[i][j][0];

            integerBoard[i][j]=c-'1';

            s.insert({i,j});

            row[i][integerBoard[i][j]]=1;

            col[j][integerBoard[i][j]]=1;

            box[i/3\*3+j/3][integerBoard[i][j]]=1;

        }

    }

    sudoku(integerBoard,0,0,s,row,col,box);

    for(int i=0;i<board.size();i++){

        for(int j=0;j<board.size();j++){

            integerBoard[i][j]++;

            board[i][j]=to\_string(integerBoard[i][j]);

        }

    }

    for(int i=0;i<board.size();i++){

        for(int j=0;j<board.size();j++){

            cout<<board[i][j]<<" ";

        }

        cout<<endl;

    }

*// printVectorOfVector(integerBoard);*

    return;

}

int main(){

    ios\_base::sync\_with\_stdio(false);

    cin.tie(0);

    vector<vector<string>>board={{"5","3",".",".","7",".",".",".","."},

    {"6",".",".","1","9","5",".",".","."},

    {".","9","8",".",".",".",".","6","."},

    {"8",".",".",".","6",".",".",".","3"},

    {"4",".",".","8",".","3",".",".","1"},

    {"7",".",".",".","2",".",".",".","6"},

    {".","6",".",".",".",".","2","8","."},

    {".",".",".","4","1","9",".",".","5"},

    {".",".",".",".","8",".",".","7","9"}};

    solveSudoku(board);

    return 0;

}