//Solving Sudoku using backtracking

#include <stdio.h>

#define SIZE 9

int matrix[9][9] = {

   {6,5,0,8,7,3,0,9,0},

   {0,0,3,2,5,0,0,0,8},

   {9,8,0,1,0,4,3,5,7},

   {1,0,5,0,0,0,0,0,0},

   {4,0,0,0,0,0,0,0,2},

   {0,0,0,0,0,0,5,0,3},

   {5,7,8,3,0,1,0,2,6},

   {2,0,0,0,4,8,9,0,0},

   {0,9,0,6,2,5,0,8,1}

};

//function to print sudoku

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");

   }

}

//function to check if all cells are assigned or not

//if there is any unassigned cell

//then this function will change the values of

//row and col accordingly

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)

           {

               \*row = i;

               \*col = j;

               //there is one or more unassigned cells

               num\_unassign = 1;

               return num\_unassign;

           }

       }

   }

   return num\_unassign;

}

//function to check if we can put a

//value in a paticular cell or not

int is\_safe(int n, int r, int c)

{

   int i,j;

   for(i=0;i<SIZE;i++)

   {

       if(matrix[r][i] == n)

           return 0;

   }

   //checking column

   for(i=0;i<SIZE;i++)

   {

       //there is a cell with the value equal to i

       if(matrix[i][c] == n)

           return 0;

   }

   //checking sub matrix

   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;d

       }

   }

   return 1;

}

int solve\_sudoku()

{

   int row;

   int col;

   //if all cells are assigned then the sudoku is already solved

   //pass by reference because number\_unassigned will change the values of row and col

   if(number\_unassigned(&row, &col) == 0)

       return 1;

   int n,i;

   //number between 1 to 9

   for(i=1;i<=SIZE;i++)

   {

       if(is\_safe(i, row, col))

       {

           matrix[row][col] = i;

           //backtracking

           if(solve\_sudoku())

               return 1;

           matrix[row][col]=0;

       }

   }

   return 0;

}

int main()

{

   if (solve\_sudoku())

       print\_sudoku();

   else

       printf("No solution\n");

   return 0;}

