Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year:2022), B.Sc. in CSE (Day)
Course Title: Algorithm Lab

Course Code: CSE 204

Section: DA
Lab Project Name: Sudoku Solver and Checker.

Student Details

	Name	ID
1.	Md. Rahul Islam Joy	212902070

Submission Date : 09/01/23
Course Teacher's Name : Dr. Faiz Al Faisal

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks: ..... **Signature:** .....

Comments: ..... **Date:** .....

Table of Contents

Chapter 1

Introduction

1. Introduction

Sudoku game is the Japanese popular game. Now a days, Sudoku puzzles are becoming popular among all over the world. The game has become popular now in a large number of countries and many developers have tried to generate even more complicated and more interesting puzzles. Now a days, we can see this game appears in almost every newspaper, in books and in many websites. This means that the algorithm is implemented based on human perceptions. I implement this game that's mean sudoku game solver and checker in C language using Backtracking.

2. Design Goals

The goal of sudoku is simple: **fill in the numbers 1-9 exactly once in every row, column, and 3x3 region**. For example, look at the above puzzle and compare it to the solved version below. Notice that every row, column and 3x3 region contain every number from 1-9 exactly once.

Chapter 2

Implementation of the project

2.1 Problem Implementation:

Given a 9 x 9 the rows and column of the Problem.

2.1.1.

```
{4,3,8,1,9,2,7,6,5},
{9,7,1,8,5,6,3,4,2},
{0,6,0,0,0,0,0,0,1},
{8,2,0,0,1,0,0,0,0},
{1,4,0,5,0,7,0,2,6},
{0,0,0,0,8,0,0,3,9},
{5,0,0,0,0,0,0,1,0},
{0,8,4,9,2,0,0,5,0},
{0,0,0,3,7,0,4,0,0}
```

2.2 Source Code

```
#include <stdio.h>
#include<conio.h>
#define SIZE 9

//sudoku problem
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}**/
   {4,3,8,1,9,2,7,6,5},
   {9,7,1,8,5,6,3,4,2},
   {0,6,0,0,0,0,0,0,1},
   {8,2,0,0,1,0,0,0,0},
   {1,4,0,5,0,7,0,2,6},
   {0,0,0,0,8,0,0,3,9},
   {5,0,0,0,0,0,0,1,0},
   {0,8,4,9,2,0,0,5,0},
   {0,0,0,3,7,0,4,0,0}

};

//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");
   }
   printf("Thank you || Press enter for exit.");
}

//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++)
      {
         //cell is unassigned
         if(matrix[i][j] == 0)
         {
            //changing the values of row and col
            *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;
    //checking in row
    for(i=0;i<SIZE;i++)
    {
        //there is a cell with same value
        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;
        }
    }
    return 1;
}

//function to solve sudoku
//using backtracking
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 we can assign i to the cell or not
        //the cell is matrix[row][col]
        if(is_safe(i, row, col))
        {
            matrix[row][col] = i;
            //backtracking
            if(solve_sudoku())
```

```
            return 1;
        //if we can't proceed with this solution
        //reassign the cell
        matrix[row][col]=0;
      }
   }
   return 0;
}
void option()
{
   printf("\n\nClick 1 for enjoy suduko solver & checker.\n");
   printf("\nIf you don't want to play more then enter 0.\n\n");
      int n;
      printf("Enter your choice : ");
      scanf("%d",&n);
      if(n==1)
      {
         if (solve_sudoku())
         print_sudoku();




         else
         printf("No solution.\n");
      }
      else if(n==0)
      {
         printf("\nThank You || Press enter for exit.");
         return 0;
      }
      else
      {
```

```
            printf("Invalid Option. Please try again.");
            option();
        }
}

int main()
{
    printf(".....Welcome.....");
    option();
    getch();
}
```

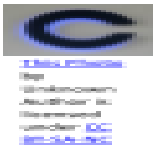
Chapter 3

Performance Evaluation

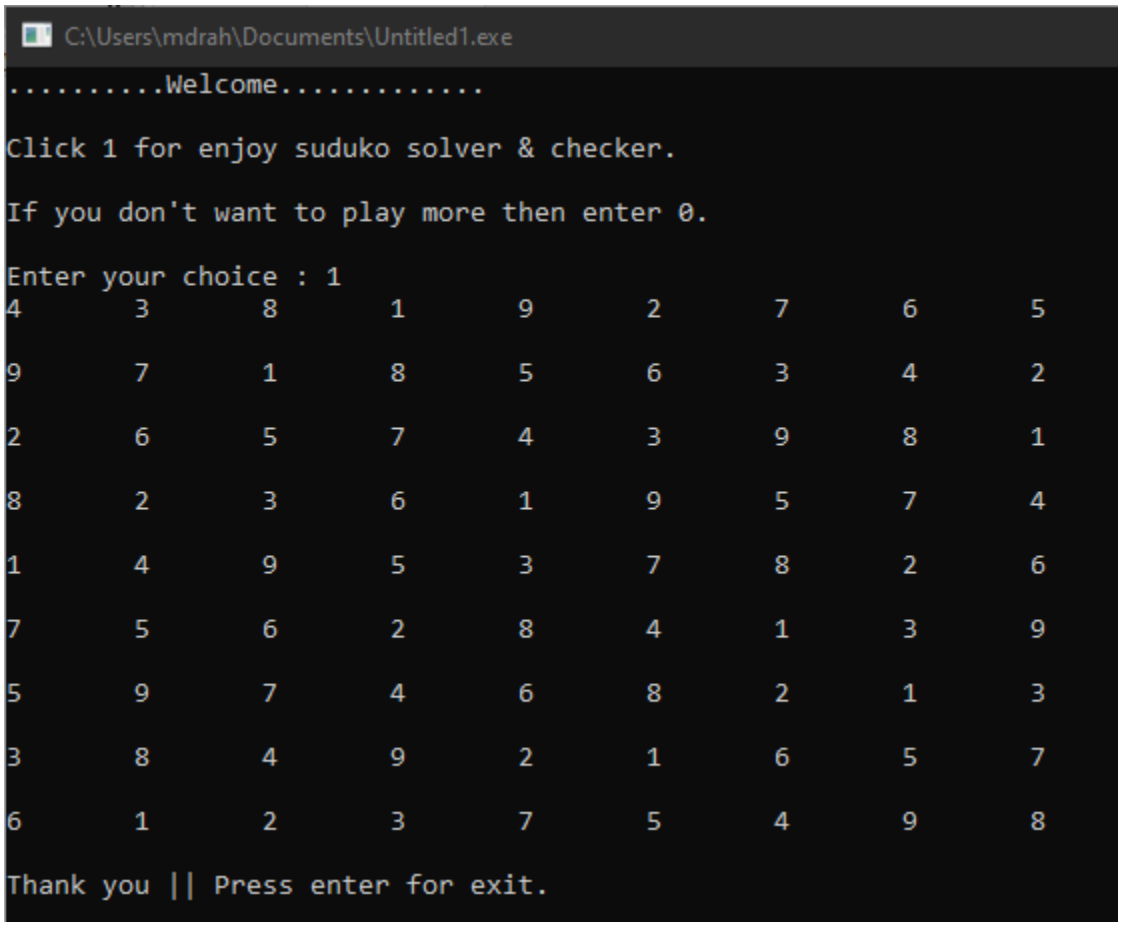
3.1 Simulation Environment:

IDE: CodeBlocks

Programming Language: C



3.2 Results and Discussions:



Sudoku – The Rules

- Only use the numbers 1 to 9 in a game of Sudoku. ...
- Avoid trying to guess the solution to the Sudoku puzzle. ...
- Only use each number once – do not repeat any numbers. ...
- Use the process of elimination as a tactic. ...
- Use cross-hatching and penciling-in techniques to solve the puzzle.

Chapter 4

Conclusion

4.1 Introduction

In this project I have used c programing language. I have successfully completed the code implementation using backtracking. The code run according to the loops and functions implemented inside the code.

4.1.1 Practical Implications

For Brain exercise. Backtracking is used in **computer science and industrial applications, load balancing in a multiprocessor computer.**

4.2 Scope of Future Work

In future by adding more features it can be more attractive game.

Source Code:

[1] GitHub: https://github.com/rahul-joy/SudukoSolver

[2] Website: https://rahulrxp.blogspot.com

References:

[1] https://rahulrxp.blogspot.com

[2] YouTube

[3] Geek for Geeks