**1. TITLE OF THE PROJECT:**

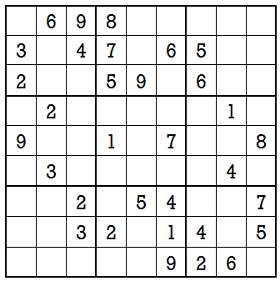
**Sudoku Game**

**2. STATEMENT ABOUT THE PROBLEM:**

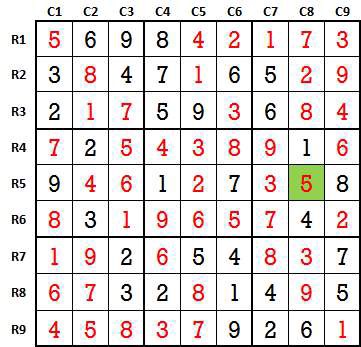
**1. Sudoku :**

Through the course of this article, we are going to explore the principle of backtracking by using it to solve a game of Sudoku. It is a logic-based puzzle game that requires you to combine numbers to come to fill the board. The objective is to fill a 9x9 grid so that each column, each row, and each of the nine 3x3 subgrids that compose the grid contain all the digits from 1–to 9. When starting, some of the grids are partially filled in.

Here is an example of a starting board.



And here is the solution to the above board. Notice that every row and every column have the digits from 1 to 9.



**2. Reason and Motivation to choose the topic:**

In our project, we have used a backtracking algorithm which is a very important skill set to solve recursive solutions. Backtracking is one of the techniques that can be used to solve the problem. We can write the algorithm using this strategy. It uses the Brute force search to solve the problem, and the brute force search says that for the given problem, we try to make all the possible solutions and pick out the best solution from all the desired solutions. This rule is also followed in dynamic programming, but dynamic programming is used for solving optimization problems. In contrast, backtracking is not used in solving optimization problems. Backtracking is used when we have multiple solutions, and we require all those solutions.

Backtracking name itself suggests that we are going back and coming forward; if it satisfies the condition, then return success, or else we go back again. It is used to solve a problem in which a sequence of objects is chosen from a specified set so that the sequence satisfies some criteria.

**3. Objective and Scope of the Project:**

The objective of the proposed project is to increase the thinking capability. The game has all the records which you perform in playing, you can make your sudoku and at any step, you can go back to one step as well as you can see the solution of it. The game of sudoku helps to increase mental thinking, vision, etc.

**4. Methodology :**

**CODE:**

#include<iostream>

#include <cstdio>

using namespace std;

#include<math.h>

#define MAX 25

int takeInput(int sudoku[MAX][MAX])

{

int size = 0;

do

{

cout<<"Choose the size of board: ";

cin>>size;

}while(size>25);

cout<<"Enter the Initial "<<size<<" X "<<size<<" Sudoku Board:"<<endl;

cout<<"Instruction: Enter 0 for blank"<<endl;

for(int i=0;i<size;i++)

for(int j=0;j<size; j++)

cin>>sudoku[i][j];

return size;

}

void displaySolution(int sudoku[MAX][MAX], int size)

{

for(int i=0;i<size;i++)

{

for(int j=0;j<size; j++)

printf("%4d", sudoku[i][j]);

cout<<endl;

}

cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

}

bool isFull(int sudoku[MAX][MAX], int size)

{

int i,j;

for(i=0;i<size;i++)

for(j=0;j<size;j++)

if(!sudoku[i][j])

return false;

return true;

}

int findPossibleValues(int sudoku[MAX][MAX], int size, int a[], int r, int c)

{

int n=0;

int i,j;

int s=(int)(sqrt(size));

int b[MAX+1]={0};

for(i=0; i<size; i++)

b[sudoku[r][i]]=1;

for(i=0; i<size; i++)

b[sudoku[i][c]]=1;

r=(r/s)\*s, c=(c/s)\*s;

for(i=r; i<r+s; i++)

for(j=c; j<c+s;j++)

b[sudoku[i][j]]=1;

for(i=1;i<=size; i++)

if(!b[i])

a[n++]=i;

return n;

}

void SolveSudoku(int sudoku[MAX][MAX], int size, bool &solved\_flag, bool &enough)

{

int i,j, a[MAX+1]={0}, n=0;

if(enough)

return;

if(isFull(sudoku, size))

{

if(!solved\_flag)

cout<<"Sudoku Solved Successfully!"<<endl;

solved\_flag = 1;

displaySolution(sudoku, size);

char more;

cout<<"Do you want to again solve this?"<<endl;

cout<<"Press 1 for 'YES', anything OTHERWISE:\t";

cin>>more;

cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

if(more != '1')

enough = true;

return;

}

int break\_flag = 0;

for(i=0;i<size;i++)

{

for(j=0;j<size;j++)

if(!sudoku[i][j])

{

break\_flag = 1;

break;

}

if(break\_flag)

break;

}

n = findPossibleValues(sudoku, size, a, i, j);

for(int l=0;l<n;l++)

{

sudoku[i][j]=a[l];

SolveSudoku(sudoku, size, solved\_flag, enough);

}

sudoku[i][j]=0;

}

int main()

{

int sudoku[MAX][MAX] = {0}, size;

size = takeInput(sudoku);

bool solved\_flag = 0;

bool enough = false;

cout<<"Finding Solutions!\n\n"<<endl;

SolveSudoku(sudoku, size, solved\_flag, enough);

if(!solved\_flag)

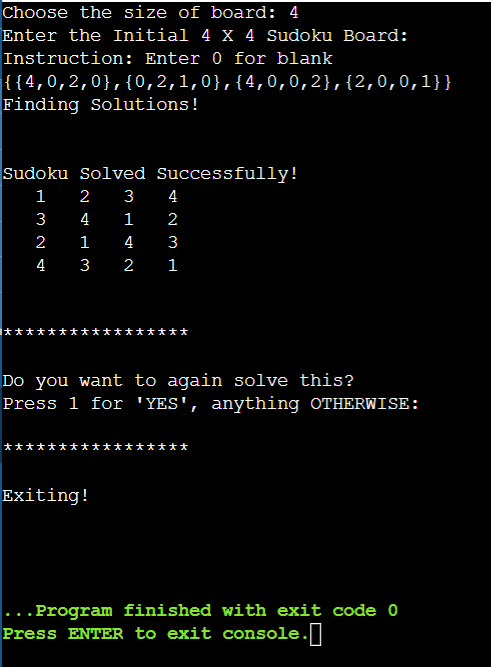
cout<<"Invalid Board!"<<endl;

cout<<"Exiting!\n\n"<<endl;

return 0;

}

**OUTPUT:-**



**5. Hardware and Software Required:**

SUDOKU recommended requirements

Memory: 512 MB

Graphics Card: Intel 82945G Express

CPU: Intel Core i3-530

File Size: 128 MB

OS: Windows XP, 7, Vista, 8, 8.1, 10

SUDOKU minimum requirement:

Memory: 256 MB

Graphics Card: Intel 82945G Express

CPU: Intel Celeron

File Size: 128 MB

OS: Windows XP, 7, Vista, 8, 8.1, 10