OOPS PROJECT

SUBMITED BY-KULDEEP(CO19336)

- HARDIK(CO19324)

- KUNAL(CO19337)

SODUKO SOLVER

What is SODUKO :

1								3
		7	2	6		4	8	
4			9	3	5			6
	3		4	8		2		
	4	1	6		9	3		
		6				8	9	
5	7	8		4				2
			3				7	
2								5

Is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 subgrids that compose the grid (also called “boxes”, “blocks”, or “regions”) contain all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution.

ALGORITHMS FOR SOLVING SODUKO-

1. Naive alogrithm-

Enter Brute Force Search (BFS). BFS is one of the most general and basic techniques for searching for possible solutions to a problem. The idea is that we want to generate every possible move within the game and then test to see whether it solves our problem. The step-by-step process is the following:

1. Generate a possible move that follows the rules of the game and has **not** been tested yet.

2. Test to see if that move wins the game/is a solution.

3. If the move wins the game, exit since you have found your solution! If the move does not win the game, add it to the list of attempted moves so you don't attempt it again.

2. BACKTRACKING-

Backtracking is an algorithm for finding all (or some) of the solutions to a problem that incrementally builds candidates to the solution(s). As soon as it determines that a candidate cannot possibly lead to a valid solution, it abandons the candidate. Backtracking is all about choices and consequences.

Abandoning a candidate typically results in visiting a previous stage of the problem-solving-process. This is what it means to "backtrack"—visit a previous stage and explore new possibilities from thereon.

PROJECT DESCRIPTION-

algorithm description-

1. Create a function that checks after assigning the current index the grid becomes unsafe or not. Keep Hashmap for a row, column and boxes. If any number has a frequency greater than 1 in the hashMap return false else return true; hashMap can be avoided by using loops.
2. Create a recursive function which takes a grid.
3. Check for any unassigned location. If present then assign a number from 1 to 9, check if assigning the number to current index makes the grid unsafe or not, if safe then recursively call the function for all safe cases from 0 to 9. If any recursive call returns true, end the loop and return true. If no recursive call returns true then return false.
4. If there is no unassigned location then return true.

Library files used-

1. iostream-for handling input and output for user.

2. conio.h-for controlling flow of programme by function like getch().

3. fstream-to load and Handel puzzle file from hard drive saved by the user for solving.

SOURCE CODE-

```
#include<iostream>
#include<fstream>
#include<conio.h>
using namespace std;

void show(int arr[9][9]){
    cout<<"\nTHE SOLUTION OF THIS PUZZLE IS : \n\n";
    for(int i=0;i<9;i++){
           for(int j=0;j<9;j++){
                  cout<<arr[i][j]<<" ";
           }cout<<"\n";
    }
}

bool check_empty_spaces(int arr[9][9],int l[]){
    for(int i=0;i<9;i++)
           for(int j = 0;j<9;j++)
                  if (arr[i][j]==0){
                         l[0]=i;
                         l[1]=j;
                         return true;
                  }
    return false;
}

bool row_check(int arr[9][9],int num,int xpos){
    for(int i=0;i<9;i++){
           if(arr[xpos][i]==num){
                  return false;
           }
    }
    return true;
}

bool column_check(int arr[9][9],int num,int ypos){
    for(int i=0;i<9;i++){
           if(arr[i][ypos]==num){
                  return false;
           }
    }
    return true;
}

bool box_check(int arr[9][9],int num,int xpos, int ypos){
    for(int i=0;i<3;i++){
           for(int j=0;j<3;j++){
```

```
                if(arr[i+xpos][j+ypos]==num){
                        return false;
                }
        }
    }
    return true;
}

bool overall_check(int arr[9][9],int num,int xpos,int ypos){
    return (row_check(arr,num,xpos)&&column_check(arr,num,ypos)&&box_check(arr,num,xpos-
    xpos%3,ypos-ypos%3));
}


bool solve(int arr[9][9]){
    int xpos,ypos;
    int l[2] = {0,0};
    if(!check_empty_spaces(arr,l)){
        show(arr);
        return true;
    }
    xpos = l[0];
    ypos = l[1];
    for(int num=1;num<10;num++){
        if (overall_check(arr,num,xpos,ypos)){
                arr[xpos][ypos]=num;
                if(solve(arr)){
                        return true;
                }
                arr[xpos][ypos]=0;
        }
    }
    return false;
}

bool input_puzzle(char filename[50],int arr[9][9]){
    ifstream f(filename);
    if(f==NULL){
        return false;
    }
    string str;
    for(int i=0;i<9;i++){
        getline(f,str);
        for(int j=0;j<9;j++){
                arr[i][j]=(str[j]-48);
        }
    }
    f.close();
    return true;
}

void puzzle_show(int arr[9][9]){
    cout<<"THE GIVEN PUZZLE IS :\n\n";
    for(int i=0;i<9;i++){
        for(int j=0;j<9;j++)
                cout<<arr[i][j]<<" ";
        cout<<"\n";
    }
}

int main(){
```

```
char filename[50];
cout<<"\n\t\t\tSUDOKU PUZZLE SOLVER\n made by
:\n\nKUNAL(CO19337)\nKULDEEP(CO19336)\nHARDIK(CO19324)\n\n\nENTER NAME OF TEXT FILE WHERE
PUZZLE IS PRESENT : ";
cin>>filename;
cout<<"\n\n";
int arr[9][9];
int recovery[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}
};
if(!input_puzzle(filename,arr)){
       for(int i=0;i<9;i++)
              for(int j=0;j<9;j++)
                     arr[i][j] = recovery[i][j];
}
puzzle_show(arr);
if(!(solve(arr))){
       cout<<"no solution possible";
}
getch();
}
```

TEXT FILE FORMAT-

The program will ask for name of text file where the unsolved puzzle is saved . for example if puzzle is saved in text file puzzle.txt , we will type puzzle.txt and the solution of the that puzzle will be displayed .

The puzzle must me saved in the format as discripted below:

1. the empty spaces will be denoted by integer 0 .
2. no other things will be typed in the file except the nine lines of the puzzle

for example

530070000
600195000
098000060
800060003
400803001
700020006
060000280
000419005
000080079