```
 /*
    This is lab I did for my data structures class where I used recursion and backtracing
  */
#include "sudoku.h"
#include <iostream>

using namespace std;

Sudoku::Sudoku(int puzzle[9][9]) {
  for(int i = 0; i < 9; i++) {
    for(int j = 0; j < 9; j++) {
      board[i][j] = puzzle[i][j];
    }
  }
}

Sudoku::~Sudoku() {
}

void Sudoku::verify() {
  for(int i = 0; i < 9; i++) {
    for(int j = 0; j < 9; j++) {
      if(!isValid(i, j)) {
        cout << "INVALID PUZZLE" << endl;
        return;
      }
    }
  }
  cout << "VALID PUZZLE" << endl;
}

void Sudoku::print() {
  for (int row=0; row<9; row++) {
      if (row % 3== 0) {
      std::cout << "-------------------------------" << std::endl;
    }

    for (int col=0; col<9; col++) {
      if (col % 3 == 0) {
        std::cout << "|";
      }

        if (board[row][col] != 0) {
        std::cout << " " << board[row][col] << " ";
      } else {
        std::cout << " . ";
      }

    }

    std::cout << "|" << std::endl;
  }
  std::cout << "-------------------------------" << std::endl;
}

bool Sudoku::isValid(int row, int col) {


  int value = board[row][col];

  for (int i=0; i<9; i++) {
    if (i == row) {
    continue;
    }

    int temp = board[i][col];
    if (temp == value) {
      return false;
    }
  }

  for (int i=0; i<9; i++) {
      if (i == col) {
      continue;
    }
```

```
    int temp = board[row][i];
    if (temp == value) {
      return false;
    }
  }

  int box_row = row / 3;
  int box_col = col / 3;

  for (int i=box_row * 3; i < box_row * 3 + 3; i++) {
    for (int j=box_col * 3; j < box_col * 3 + 3; j++) {
      if (i == row && j == col) {
        continue;
      }

      int temp = board[i][j];
      if (temp == value) {
        return false;
      }
    }
  }

  return true;
}

void Sudoku::solve(){
  solveHelper(0, 0);
}



bool Sudoku::solveHelper(int row, int col) {
  if (row == 9) {
  return true;
  }
  if (board[row][col] == 0) {
    for (int i = 1; i<=9; i++) {
      board[row][col] = i;
      if(isValid(row,col)) {
        if(col == 8) {
          if (solveHelper(row+1,0)) {
            return true;
          }
        }
        else {
          if(solveHelper(row,col+1)) {
            return true;
          }
        }
      }
    }
    board[row][col] = 0;
  }
  else {
    if(col == 8) {
      return solveHelper(row+1,0);
    } else {
      return solveHelper(row, col+1);
    }
  }
  return false;
}
```