Nikolas Sanderson
Nicole Contreras

This sudoku solver uses recursive backtracking to solve sudoku puzzles imputed from the provided txt file. Here in figure 1, is the main function which implements recursion. The function initially starts with the base case of seeing if there are any empty squares. If there are none then the sudoku is solved and the function returns true. If the base condition is not filled however the function then finds an empty slot and begins with placing 1. It then checks if the 1 value conflicts with the row, column, and square it's in. If it sees no conflicts the number is placed at the cell and the function is called again. This is where recursion occurs and the base case is again tested. If there is a conflict the value at the index of the sudoku board is reset to blank and another value up to and including 9 is tested. IT continues backtracking until it can again move forward testing different values.

```
if(!findempty(row,col))
        return true;
    for (int num = 1; num<= 9; num++)
    {
        if (isValid(row,col,num))
        {
            setCell(row,col,num);
            if(solveSudoku())
                return true;
            clearCell(row,col);
        }

    }
    return false;
```

Figure 1. Sudoku Solver Code

Our function is mainly the brute force method of sudoku solving as each row, column, and square conflict is tested each time. This method is thorough however ultimately not the quickest method with an upper bound runtime of $O(n^2)$ but it comes with the most intuitive implementation for code writing. Time based estimates are around 14 minutes, with an average of 4,095,289 recursive calls with some puzzles taking millions of calls. Figure 2, shows the brute force within the isValid function.

```
for (int i = 1; i < 10; i++)
    {
            if(num == value[row][i])
            {
                return false;
            }
    }
    for (int i = 1; i < 10; i++)
    {
            if(num == value[i][col])
            {
                return false;
            }
    }
    int squareNum = squareNumber(row, col);
    for (int i = 1; i <= BoardSize; i++)
    {
            for (int j = 1; j <= BoardSize; j++)
            {
                if (squareNumber(i, j) == squareNum)
                    if (value[i][j] ==  num)
                        return false;
            }
    }

    return true;
```

Figure 2. isValid function Code