P1 – Solving Sudoku

Prep

2.1 Background Rules

For a 9 x 9 sudoku (3 x 3 x 3 x 3) the rules are as followed:

- One of each of the numbers 1 to 9 must be occupied at least once in each row, each column and each sub square (3x3).
- This means there must not be any repeated numbers in any row, column or sub square.

2.2 Algorithm Design

- One of the easiest methods is to brute force your way. This means for each empty square you should try one number, see if it fits and then if it does move on to the next, but if it runs out of numbers which can fit then you go back and change it.
- Another method which I think would be to have an array of the numbers 1 to 9 for each of the empty squares. Then go through each of the fixed numbers and then delete the value from the arrays in both the column, row and sub square. Then once one of the arrays is equal to the length of 1, then that must mean that it is the only possible value which can go there. For some complex sudoku this may need some brute force if there is a case where this algorithm gets stuck.

I think the second method is the best way at solving. This is because it requires the least amount of iteration and is also simpler to implement, both of which will contribute to the sudoku being solved faster. A pseudocode algorithm has been provided below.

```
Start
array board[81][9] = [nums(1,9), ]
numSolved = 0
solved = false

readSudokuFile

while not solved
        for i 0 to 80
                if board[i].length equals 0
                        numSolved++

                        deletePosibilityFromRow
                        deletePosibilityFromColumn
                        deletePosibilityFromSubsquare

        if numSolved equals 81
                solved = true

writeSudokuFile
printSudoku
```

2.3 File Handling

To handle files in C we use the fopen function to open it as either read or write (or append). Then we can use the fgetc function to get a character until the character is NULL when we know were at the end of the file.

To handle files in C++ we use the fstream library. We use the ifstream for reading the file and ofstream to output to the file. We can use the open function with an ifstream object to open a file and then use (ifstream) >> (variable) to read the data to the char buffer. The code can be found below.

```
#include <iostream>
#include <fstream>

using namespace std;

#define BUF_SIZE 100

char buf[BUF_SIZE];

int main() {
    ifstream file;
    file.open("test.txt");

    file >> buf;

    cout << buf << endl;

    file.close();
}
```