Hello World - 02/02/21
https://stackoverflow.com/questions/9935027/cout-does-not-name-a-type
https://stackoverflow.com/questions/28236870/undefined-reference-to-stdcout

Coding Hello world is different in C++ to C as it uses a different header.

<iostream> reads inputs and writes outputs to terminal, or to file. It has different namespaces for the commands. When using the same namespace across the entire program it is easier to include the using namespace command. When using different namespaces the command can be written as:

std::cout << “Hello world!\n”;

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!\n";
}
```

When compiling I initially had issues, a c++ program is compiled using the g++ command instad of gcc.

```
PS G:\1204_Advanced_Programming> gcc .\hello_world.cpp -o hello
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x19):
undefined reference to `std::cout'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x1e):
undefined reference to `std::basic_ostream<char, std::char_traits<char> >&
std::operator<< <std::char_traits<char> >(std::basic_ostream<char,
std::char_traits<char> >&, char const*)'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x35):
undefined reference to `std::ios_base::Init::~Init()'
C:\Users\ludde\AppData\Local\Temp\ccUEMhQQ.o:hello_world.cpp:(.text+0x56):
undefined reference to `std::ios_base::Init::Init()'
collect2.exe: error: ld returned 1 exit status
PS G:\1204_Advanced_Programming> g++ .\hello_world.cpp -o hello
PS G:\1204_Advanced_Programming> .\hello.exe
Hello world!
```

P1: Solving Sudoku
https://en.wikipedia.org/wiki/Sudoku_solving_algorithms
https://www.geeksforgeeks.org/sudoku-backtracking-7/
https://www.tutorialspoint.com/cplusplus/cpp_multi_dimensional_arrays.htm

Algorithm Design - 02/02/21

1) What are the various types of sudoku solving algorithms available? (List at least 2)

For solving sudoku there are two main methods:

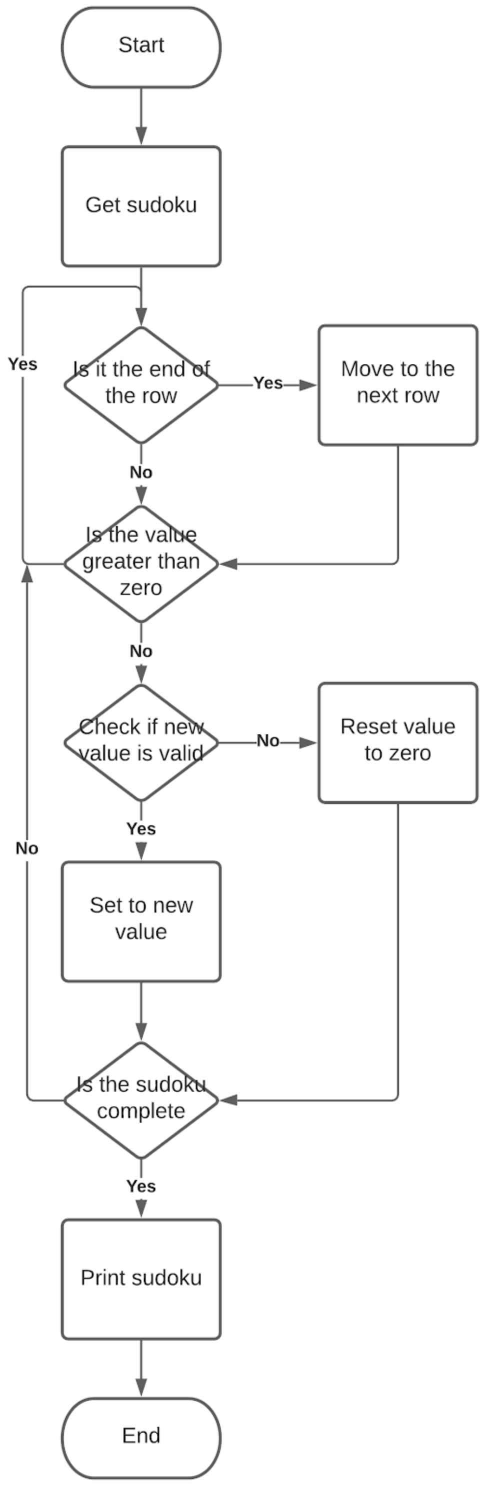
Backtracking is a type of depth-first search algorithm that is looking for the solution using recursion. It will look down the branch until an error is detected and then backtrack till the last split.

Stochastic searches randomly assigns numbers to the cells and then checks for shuffles that reduces the number of total errors. It will keep iterating this until a global maximum is reached, i.e. the sudoku is solved.

2) Work together with your partners to decide on the best approach to solve the problem, then, choose one algorithm that you plan to implement in this lab and list down its pros and cons.

I will be implementing a backtracking algorithm. The advantages of this method are the solution is guaranteed and the algorithm is simpler to implement. The disadvantage of this method is the program will be comparatively slow to other algorithms.

3) Then, draw a suitable program flowchart to illustrate your approach in your logbook. You may draw multiple flowcharts to explain your program's subroutines or functions if needed.



File Handling - 02/02/21

4) How do you read a character from file in C++?

To read from a file in C++ the <fstream> and <string> headers are needed.

The code will read in the file one word at a time and then print it out to the terminal.

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    fstream inputStream;
    string text;

    inputStream.open("lazy_doggo.txt");

    while(!inputStream.eof())
    {
        inputStream >> text;
        cout << text << " ";
    }

    return 0;
}
```

5) How do you read a character from file in C?

To read in a file using c the fgetc command is used on characters and the fgets command is used on strings.

```
#include <stdio.h>

int main()
{
    FILE *fp;

    fp = fopen("test.txt", "r");
    while (!feof(fp))
    {
        char letter = fgetc(fp);
        printf("%s", letter);
    }
    fclose(fp);
}
```

6) Will your answer for 5) work in a C++ code?

My code for part 5 should work in C++ as all C features should work. However, it is recommended against using C libraries as there is the potential to conflict with C++ libraries.

File Reading - 03/02/21

To read in the file I used the same method as in my preparation. However, I also added some processing to convert the value read in from a character to an integer value.

```
int *import_board(void)
{
    fstream inputStream;
    char input;
    int number;
    int row = 0;
    int col = 0;
    inputStream.open("SUDOKU.txt");

    while (!inputStream.eof())
    {
        inputStream >> input;

        //convert character into number
        number = input - 48;
        if (number < 0 || number > SIZE)
        {
            number = 0;
        }

        //update row position
        if (col == SIZE)
        {
            row++;
            col = 0;
        }

        //set position to number
        grid[row][col] = number;
        col++;
    }

    return (int *)grid;
}
```

Sudoku Display - 03/02/21

To print out the board that I read in I created a function, display_board(). display_board() also creates a grid around the sub-boards. This is an aesthetic change, but it will make it easier for the user of the program to interpret the sudoku board.

```
void display_board(int *board)
{
    for (int row = 0; row < SIZE; row++)
    {
        if (row % SUBSIZE == 0)
        {
            cout << " ------- ------- ------- " << endl;
        }

        for (int col = 0; col < SIZE; col++)
        {
            if (col % SUBSIZE == 0)
            {
                cout << "| ";
            }

            int boardPos = (SIZE * row + col);
            if (board[boardPos] == 0)
            {
                cout << "_ ";
            }
            else
            {
                cout << board[boardPos] << " ";
            }
        }
        cout << "|" << endl;
    }

    cout << " ------- ------- ------- " << endl;
}
```

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.2_Display> g++
.\Sudoku_solver.cpp .\Sudoku_solver.h -o sudoku
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.3_Display> .\sudoku.exe
 ------- ------- -------
| _ 2 _ | 8 _ _ | _ _ _ |
| 9 _ 1 | _ _ _ | _ _ _ |
| _ 7 _ | 2 4 6 | _ _ _ |
 ------- ------- -------
| 2 _ 9 | _ _ 5 | _ _ 4 |
| _ _ 5 | _ 8 _ | _ 6 _ |
| _ _ _ | _ 6 1 | _ _ _ |
 ------- ------- -------
| _ _ _ | _ _ _ | _ 5 7 |
| 3 _ _ | _ _ _ | 8 _ _ |
| _ _ _ | 3 2 7 | _ _ _ |
 ------- ------- -------
```

Solving Sudoku - 03/02/21

```
#define SUBSIZE 3
#define SIZE 9
int grid[SIZE][SIZE] = {0};

int main()
{
    int *board = import_board();
    display_board(board);
    cout << endl;

    if (solve_sudoku(board, 0))
    {
        display_board(board);
    }
    else
    {
        cout << "you're sudoku board has no solutions!" << endl;
    }

    return 0;
}
```

To solve the sudoku I implemented two functions. The first was a recursive function that would enable the search through the different branches of the sudoku.

When initially implementing this function I found that I had the issue that my condition to check if the sudoku was complete was in the wrong place so I moved this up from the bottom (where it was on my flowchart) to the top.

```
bool solve_sudoku(int *board, int boardPos)
{
    int row = boardPos / SIZE;
    int col = boardPos % SIZE;

    //is the sudoku complete
    if (row == SIZE)
    {
        return true;
    }

    //is it the end of the row
    if (col == SIZE)
    {
        row++;
        col = 0;
    }

    //is the value greater than zero
    if (board[boardPos] > 0)
    {
        if (solve_sudoku(board, boardPos + 1))
        {
            return true;
        }
    }
    //check if new value is valid
    for (int number = 1; number <= SIZE; number++)
    {
        if (valid_number(board, row, col, number))
        {
            board[boardPos] = number;
            if (solve_sudoku(board, boardPos + 1))
            {
                return true;
            }
        }
    }

    board[boardPos] = 0;
    return false;
}
```

The second function would be called from the solver to check for a valid number to fill the position in the board with.

```c
bool valid_number(int *board, int row, int col, int num)
{
    //check if row is valid
    for (int i = 0; i < SIZE; i++)
    {
        if (board[(SIZE * row) + i] == num)
        {
            return false;
        }
    }

    //check if column is valid
    for (int i = 0; i < SIZE; i++)
    {
        if (board[(SIZE * i) + col] == num)
        {
            return false;
        }
    }

    //check if sub-board is valid
    int startRow = row - row % SUBSIZE;
    int startCol = col - col % SUBSIZE;
    for (int i = 0; i < SUBSIZE; i++)
    {
        for (int j = 0; j < SUBSIZE; j++)
        {
            if (board[(startRow + i) * SIZE + startCol + j] == num)
            {
                return false;
            }
        }
    }

    return true;
}
```

My first implementation of the sub-board check did not work. I had to change the code from the one below, to the final code

```
    //check if sub-board is valid
    int sub_row = row - row % subsize;
    int sub_col = col - col / subsize;
```

Once the changes, and some minor bug fixing was completed I was able to compile and run my program so that it was able to solve the sudoku.

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku\3.3_Sudoku_solver>
.\sudoku.exe
 ------- ------- -------
| _ 2 _ | 8 _ _ | _ _ _ |
| 9 _ 1 | _ _ _ | _ _ _ |
| _ 7 _ | 2 4 6 | _ _ _ |
 ------- ------- -------
| 2 _ 9 | _ _ 5 | _ _ 4 |
| _ _ 5 | _ 8 _ | _ 6 _ |
| _ _ _ | _ 6 1 | _ _ _ |
 ------- ------- -------
| _ _ _ | _ _ _ | _ 5 7 |
| 3 _ _ | _ _ _ | 8 _ _ |
| _ _ _ | 3 2 7 | _ _ _ |
 ------- ------- -------


 ------- ------- -------
| 4 2 3 | 8 1 9 | 5 7 6 |
| 9 6 1 | 5 7 3 | 2 4 8 |
| 5 7 8 | 2 4 6 | 1 3 9 |
 ------- ------- -------
| 2 1 6 | 7 3 5 | 9 8 4 |
| 7 4 5 | 9 8 2 | 6 1 3 |
| 8 3 9 | 1 6 4 | 7 5 2 |
 ------- ------- -------
| 1 9 2 | 4 5 8 | 3 6 7 |
| 3 8 7 | 6 9 1 | 4 2 5 |
| 6 5 4 | 3 2 7 | 8 9 1 |
 ------- ------- -------
```

Optional Additional Work - 03/02/21

As my work was fully parameterised I was able to easily change it so that it would solve the sudoku for a 16x16 board.

```
#define SUBSIZE 4
#define SIZE 16
```

Unfortunately, in the time I was not able to modify the sudoku so that it was easily completable. However, I was able to successfully read and print the sudoku.

I found that the only necessary change to the working of the code was that the horizontal box lines were not parameterised.

```
PS G:\1204_Advanced_Programming\P1_Solving_Sudoku> .\su.exe
 ------- ------- -------
| _ 2 _ 8 | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| 9 _ 1 _ | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| _ 7 _ 2 | 4 6 _ _ | _ _ _ _ | _ _ _ _ |
| 2 _ 9 _ | _ 5 _ _ | 4 _ _ _ | _ _ _ _ |
 ------- ------- -------
| _ _ 5 _ | 8 _ _ 6 | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | 6 1 _ _ | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | _ _ _ 5 | 7 _ _ _ | _ _ _ _ |
| 3 _ _ _ | _ _ 8 _ | _ _ _ _ | _ _ _ _ |
 ------- ------- -------
| _ _ _ 3 | 2 7 _ _ | _ _ _ _ | _ _ _ _ |
| _ 2 _ 8 | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| 9 _ 1 _ | _ _ _ _ | _ _ _ _ | _ _ _ _ |
| _ 7 _ 2 | 4 6 _ _ | _ _ _ _ | _ _ _ _ |
 ------- ------- -------
| 2 _ 9 _ | _ 5 _ _ | 4 _ _ _ | _ _ _ _ |
| _ _ 5 _ | 8 _ _ 6 | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | 6 1 _ _ | _ _ _ _ | _ _ _ _ |
| _ _ _ _ | _ _ _ 5 | 7 _ _ _ | _ _ _ _ |
 ------- ------- -------
```