1. Description of Project Concept Programs

1.1 Simple and colourful GUI with an active button:

- This program will prove that I am able to create and code a working and correctly designed user interface that fulfills all of the necessary specifications that were set out. This will contain the simple buttons and forms the program will need.
- I will be using the swing library that will give me a much easier way to create the interface and maybe some third party software to make the process more user friendly.
- This will allow the basic ideas for what is needed within the GUI and what processes are best within them.

The SVN Location:
https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Concept%20Programs/GUI/tags/

Main Method: SudokuGUI.java

1.2 Data structures: Dynamic Trees and Priority Queue's; populated with large random data sets.

- The program will consist of ways to exercise and use data structures that will need to be used in the final version by using dynamic trees to filter and sort information. For example, a priority queue, which is a way of storing data the data in a heap.
- The data structure program will allow a familiar base to look at once the Sudoku solver is under way as these are data structures that I have not used often and it was necessary to program them and understand the way they work.

SVN Location:
https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Concept%20Programs/Data%20Structures/tags/

Main Method: Runner.java

1.3 Game tree for Tic-Tac-Toe

- This considers a two-player game where there are very limited options. This uses a game tree, which is a way of displaying every option that a user could possibly use, from the highest, which displays the first option at the head and below branches into the other options of choices where to put a token. MiniMax algorithm is used to judge what the next user can do next and then this is then restricted into only the valid boards, i.e. When a player wins the tree does not need to continue, this is called pruning.
- This MiniMax algorithm is used to take the initial board and then iterate through the other possible boards that are valid, these are then scored and the algorithm will return the highest scoring game move. This is very relatable to the Sudoku puzzle about the best number to return.

SVN Location:

https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Concept%20Programs/TicTacToe/tags/

Main Method: TicTacToe.java

1.4 Eight Queens using Backtracking

- Using the backtracking algorithm, in this case, means placing a queen in a column and checking for clashes with previously placed queens. In that column if a row is found without a clash the queens mark this row/column as part of a solution, however if it returns false it is removed and then the next index is used.
- Eight Queens puzzle is done using a backtracking algorithm that allows the option to try every possible cell but try the next cell if the current one does not work. This could relate the Sudoku program due to the need to check the states of many cells, backtracking could be a good way.

SVN Location:

https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Concept%20Programs/8Queens/tags/

Main Method: 8Queens.java

2. Reports SVN Locations

2.1 Interim Report

https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Documentation/Interim Submission/Documents/InterimReport/

2.2 Term Reports

https://svn.cs.rhul.ac.uk/personal/zwac016/Year%203/Final%20Project/Documentation/Interim Submission/Documents/TermReports/