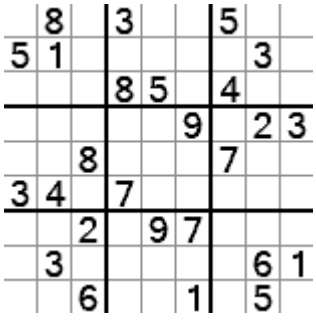
P1

Solving Sudoku

The intention of this lab is to get you to think about a problem before you start to solve it so that when you come to code it up you and your partner know exactly what to do. The name of the game is Sudoku and it's your job to solve it. The outcomes should be an idea of the importance of proper preparation and to get you to start thinking in an object-oriented way about a reasonably complex problem.



Schedule

Preparation time	:	3 hours
Lab time	:	3 hours

Items provided

Tools	:	None
Components	:	None
Equipment	:	None
Software	:	Eclipse and MinGW set up for C++ projects

Items to bring

Essentials. A full list is available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Before you come to the lab, it is essential that you read through this document and complete ***all*** of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

> **Academic Integrity** – *If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.*

Revision History

January 27 2012	Jeff Reeve (jsr)	First version of this lab created
December 18 2019	Ivan Ling (il)	Revised lab prep and lab work

1 Aims, Learning Outcomes and Outline

This laboratory exercise aims to:

- Introduce you to analysing an intricate software problem.
- Introduce you to documenting your design so that both lab partners know clearly what they are expected to do.
- Introduce you to detailing interfaces so that individually designed parts of a program fit together.

Having successfully completed the lab, you will be able to:

- Understand file streams and how they work.
- Know some of the ideas from C++ that enable you to write "safer" code.
- Design software that isolates data structures.

The overall objective of the lab is of course to solve the game Sudoku. If you aren't familiar with it, How to solve Sudoku will give you the idea (or buy a newspaper and have a go!). The rules are systematic and by following them, things should work out ok. You are NOT required to use classes in this lab – we will explicitly use this example to see how to organise this problem in an Object Oriented manner later in the lectures. The code you write will look like a normal C program, as you will notice that normal C syntaxes works fine in C++.

---

2 Preparation

Read through the course handbook statement on safety and safe working practices, and your copy of the standard operating procedure. Make sure that you understand how to work safely. Read through this document so you are aware of what you will be expected to do in the lab.

2.1 Background – The rules

Read up on the rules of sudoku and write it down in your log book. Illustrate each explanation with a simple diagram.

2.2 Algorithm Design

An algorithm is a process or a set of rules which is to be followed in a problem-solving operation. The repetitive tasks in an algorithm can usually be handled quickly and efficiently with a program.

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

(ii) 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.

(iii) 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.

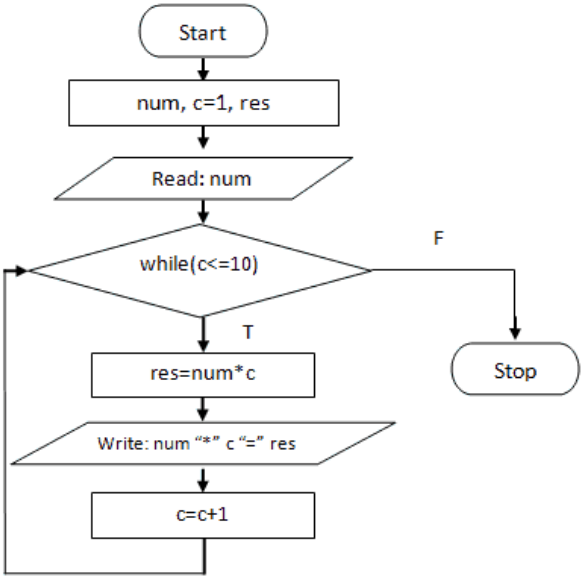


Figure 1: Example of a program flowchart

The program flowchart in your logbook should act as a guide to you when you are implementing the sudoku solver code in the lab.

2.3 File Handling

For your program to solve a sudoku puzzle, it must first read the Sudoku question files from a text file.

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

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

(iii) Will your answer for (ii) work in a C++ code?

---

3 Laboratory Work

At the start of the session make sure that you can log on to the PC and can start eclipse and set the project to a C++ one. Code file extensions are .cpp and headers are .h. Have separate header file for declarations of functions from their definitions.

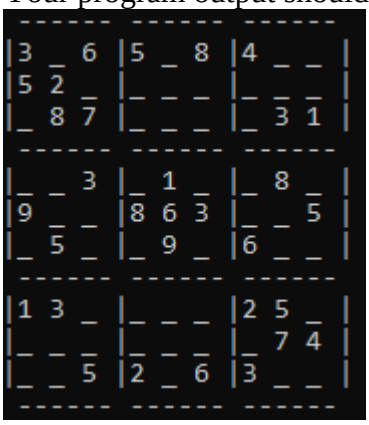
3.1 File reading

1. Using the code prepared during your lab prep, write a simple program which could read a stream of character from a SUDOKU file programmatically. The SUDOKU file is provided in the ECS lab site.
2. To see the content of the SUDOKU file, right click on it on the explorer and select "open with". Then, choose a suitable text editor software (e.g. Notepad, Notepad++) to open it. This should allow you to see the format of the SUDOKU question.
3. Edit your program such that the program reads and store the contents of the SUDOKU file into a 9X9 array.

3.2 Sudoku Display

1. Write a function "display_board()" to print out the Sudoku array.

2. The empty spaces in the SUDOKU file is represented by an 'X'. Write appropriate conditional statements to print an empty space instead of 'X' on the board.
3. Your program output should look something like this:



Where the character "_" is used to represent the empty space.

3.3 Sudoku solving

1. Implement your chosen algorithm as a function which takes the sudoku array as an input.
2. Your function should return an array containing the solution of the sudoku.
3. Print out the solution and verify it manually. You may create your own input files with sudoku puzzles found at https://www.puzzles.ca/sudoku/

---

4 Optional Additional Work

Marks will only be awarded for this section if you have already completed all of Section 3 to an excellent standard and with excellent understanding.

If it has gone good so far implement a solver for a 16x16 Grid that uses the hexadecimal numbers 0 to F with 4x4 sub grids.

Appendices

References

http://sudokuprofessor.com/?gclid=COiG1o6wiLUCFeTMtAodLTMABg Learn Sudoku.

http://cws.cengage.co.uk/vickers/students/Vickers_CH02_019-036.pdf How to think like a programmer.

http://www.cplusplus.com/reference/ios/ios_base/setf/ Set the stream to hexadecimal.