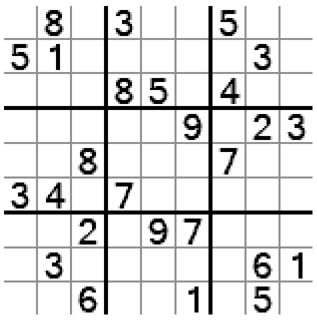
P1

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.



	8		3			5		
5	1						3	
			8	5		4		
					9		2	3
		8				7		
3	4		7					
		2		9	7			
	3						6	1
		6			1		5	

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

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 ordinary old 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

The rules of the game are simple. You are presented with a 9x9 grid of squares grouped together in 3x3 sub grids partially filled in with numbers in the range 1 to 9. The idea is to fill in the grid completely so that all the numbers in each sub grid contain the numbers 1 to 9 and each column and row of the grid also contains the numbers 1 to 9.

2.2 Design

Read these How to Think Like a Programmer pages before you start. You need to get your head together with your partner to decide how you are going to go about solving the puzzle. You might use a simple approach of working out the possible values that each square might take and then determine if you can place any of these choices according to the rules. On the other hand you might like to implement an intuitive way that you play the game yourself. Clearly any approach is iterative so you have a way of grading how difficult a puzzle is. The output from this section is a design of your solution presented in any way that you like. Flow charts, pseudo code, UML diagrams or ad hoc diagrams of your own are all acceptable. You also need to present a division of labour.

2.3 File Handling

The course text has a good section on reading a text file (Savitch5/e section 2.4). This is how you are required to input the puzzle. Use can int or char types . If you stream the file to something declared as an integer then it will be read as an integer without you needing to make any conversions. If you try the extra work however you will need to read the input as hexadecimal numbers. Use the cplusplus pages to find out how. The output is a program that can read a puzzle into a normal C array of integers.

---

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.

Subsection 1

Following your plan code up and test your Sudoku Solver.

---

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.