International Foundation Programme / Year Zero Programme
Resubmission of Coursework – following Exam Board decision
July 2017

IA160 Computer Programming

Please follow the instructions below very carefully. Do not simply create your own titles or use existing titles from the academic year. If you do, you will receive a mark of zero. You must keep a copy of all work that you submit. No extensions will be given to this deadline, and the normal late submissions policy applies

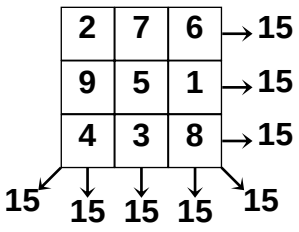
Assignment task: Magic Square Puzzle

You are required to solve and write Python code for the task below; and also produce a written report to describe and analyse the problem you are solving, explain your solution to the problem, and detail how you have tested and evaluated your solution.

You must use Python version 3 for this assignment; submissions using any other programming language or Python version will not be marked and could result in a mark of zero for this assignment.

Task (Magic Square Puzzle):

- Write a Python program to validate a given Magic Square puzzle board.
- Background:
    - A typical Magic Square is an n x n square grid filled with integers 1 to n2.
    - The integers should be placed in the grid so that each row, each column, and each of the two diagonals all add up to the same value.
    - Conditions state that no single integer can be used more than once in the grid.
    - The grid can vary in size, but must be a square formation (3x3, 4x4, 5x5, etc.).
    - If you are unsure of the rules of a Magic Square puzzle then look them up.



Examples of some valid Magic Square puzzle boards:

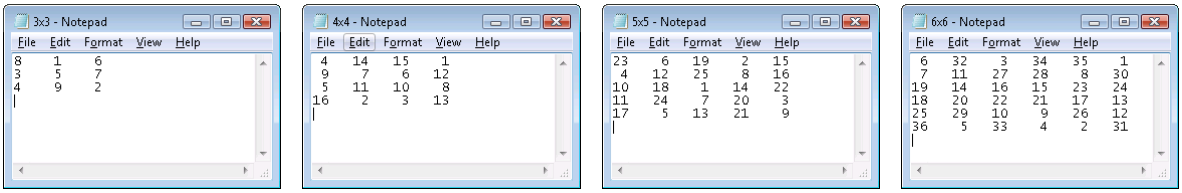
8	1	6
3	5	7
4	9	2

6	32	3	34	35	1
7	11	27	28	8	30
19	14	16	15	23	24
18	20	22	21	17	13
25	29	10	9	26	12
36	5	33	4	2	31

23	6	19	2	15
4	12	25	8	16
10	18	1	14	22
11	24	7	20	3
17	5	13	21	9

4	14	15	1
9	7	6	12
5	11	10	8
16	2	3	13

- The puzzle boards will be supplied to the program in the form of a text file, for example:



- The program should prompt the user for the filename of a text file to open – some Python error handling should be used in case the user enters the name of a file that does not exist.

- The program should read the contents of the given text file and store this in a suitable Python data structure.

- Appropriate validation should be in place to ensure the supplied file contains a complete *n* x *n* square grid of digits, and ensure the digits are all integers in the range 1 to n^2; a suitable error message should be returned where necessary.

- Assuming a suitable *n* x *n* square grid of integers is successfully loaded into the chosen data structure; further checks should then be performed to ensure that the sums of each row, each column, and each of the two diagonals are all equal.

- If any errors are detected in the puzzle board then details of these errors should be given to the user.

- If the puzzle board passes all the validation checks, then it should be written into a new text file using the same format as the original input file (see the example text files above) – the name for this new text file should be the original filename entered by the user prefixed with "VALID_".

- To make your code more efficient and neat you should write your segments of your code in functions where appropriate for particular tasks.

Report:

- Produce a written report in a Microsoft Word document (or similar) to describe your program.

- The report must contain the following sections:
  - Introduction – this should contain some background information, generally summarise the problem being solved in this task and demonstrate a clear understanding of the problems.
  - Solution Explanation – this is the main body of your report where you should describe your design and solution to the problem/task outlined in the introduction, and explain any key aspects to the program. You can also use this section to explain your design rationale for certain parts of your code (i.e. why have you done something in a certain way).
  - Testing – explain how you have tested your program, and if appropriate give some sample data and output examples from the program. There should be evidence of your program being tested given either in this section or in the appendix and referred to from this section.
  - Conclusion – use this section to evaluate your work (i.e. your solution to the problem described in the introduction), for example, what has worked well, what would you like to change next time.
  - Appendix – you should copy and paste your entire program code into this final section of the report.

- Please note, any code pasted into the report (including the appendix) should use the `Courier New` (or a similar fixed-width) font to distinguish it from any other text; screenshots must not be used for source code.

- Maximum length for the report: ~2000 words (excluding contents page, code segments, bibliography, and appendix).

Deliverables:

- Python script file for the task
- Report – Microsoft Word document (or similar)

Contact details: Dr. Ian Mothersole – imothe@essex.ac.uk
See IA160 Moodle site for office hours

Submission instructions: **Your assignment is to be submitted by email directly to ifp@essex.ac.uk**

Submission time and date: by 13:00pm GMT on Thursday 24 August 2017.

Marking Scheme

	Tasks	Marking Criteria Marks %	
Program / code	Program can successfully read the puzzle board text file into a suitable data structure. Program should prompt the user for a filename and have good error handling in case a file does not exist.	Cannot read a text file: Text file contents read successfully:	0 1-2
		No Error handling: Error handling in place for filenames:	0 1-2
		Incorrect/inappropriate data structure: Data structure functional but not efficient: Good/correct data structure used:	0 1-2 3-6
		Total: 10	
	General validation of the Magic Square puzzle board; for example, checking for a valid n x n grid, checking only integers 1 to n^2 used (and not repeated), etc. Further checks for equal sums in each row, each column, and each diagonal.	No general validation: Some general validation: Good/complete validation:	0 1-10 11-15
		No checking for equal sums: Some checking for equal sums: Extensive checks performed:	0 1-10 11-15
		Total: 30	
	Detailed reporting of any errors found in the Magic Square puzzle board. Valid Magic Square puzzle board written to text file correctly (using the correct filename and the same formatting as the original file).	No error reporting: Error report given, but not detailed: Full and detailed error report:	0 1-10 11-15
		Not written to a file: Written to a file but contains some errors: Correctly written to a file:	0 1-2 3-5
		Total: 20	
	Code must be neat and well structured and include suitable comments where appropriate. Program design should be efficient.	Code is poorly structured/not consistent and/or difficult to follow: Code is neat and well structured:	0-2 3-5
		Poor and/or inefficient program design: Good program design: Excellent	0 1-2 3-5
		Total: 10	
Report	Make sure all the sections of the report are presented with appropriate headings, and they appear in a logical sequence like suggested in this assignment brief. Any sources that you have used for this assignment must be correctly cited using the IEEE referencing style. Use clear and suitably sized fonts. A contents page should be included at the beginning and a references section at the end of the report. Your name and student number should appear clearly on the title page.	Poor document structure and formatting: Good document structure and formatting:	0-2 3-6
		Poor (or no) clear introduction: Good and clear introduction:	0-2 3-5
		Poor explanation of solution/code: Good explanation of solution/code:	0-3 4-8
		Insufficient evidence of testing: Good testing/evaluation:	0-2 3-6
		Poor (or no) clear conclusion: Good and clear conclusion:	0-2 3-5
		Total: 30	