UNIVERSITY OF SASKATCHEWAN

Department of Computer Science
176 Thorvaldson Building
110 Science Place, Saskatoon, SK, S7N 5C9, Canada
Telephine: (306) 966-4886, Facimile: (306) 966-4884

Assignment 2

Practicing your Development and Design Skills

Date Due: May 30, 2019, 7pm **Total Marks: 40**

General Instructions

- **This assignment is individual work.** You may discuss questions and problems with anyone, but the work you hand in for this assignment must be your own work.
- Each question indicates what to hand in. You must give your document the name we prescribe for each question, usually in the form aNqM, meaning Assignment N, Question M.
- Make sure your name and student number appear at the top of every document you hand in. These conventions assist the markers in their work. Failure to follow these conventions will result in needless effort by the markers, and a deduction of grades for you.
- Do not submit folders, or zip files, even if you think it will help.
- Programs must be written in Python 3.
- **Assignments must be submitted to Moodle.** There is a link on the course webpage that shows you how to do this.
- **Moodle will not let you submit work after the assignment deadline.** It is advisable to hand in each answer that you are happy with as you go. You can always revise and resubmit as many times as you like before the deadline; only your most recent submission will be graded.
- Read the purpose of each question. Read the Evaluation section of each question.

Version History

- **05/23/2019**: released to students

Question 1 (30 points):

Purpose: To practice the development process for a familiar problem that is complex enough to be interesting. You are to follow a development process, as outlined in the readings and lectures. You are expected to plan the development of this application, starting from the requirements description, given below.

Degree of Difficulty: Moderate

A *Sudoku Square* is a 9×9 square of numbers with 3 constraints. The first two constraints concern rows and columns:

- Every *row* contains all the numbers from 1 to 9 exactly once.
- Every *column* contains all the numbers from 1 to 9 exactly once.

In addition, the sudoku square has 9 *blocks* that are 3×3 in size, and there is one constraint on these blocks:

- Every *block* contains all the numbers from 1 to 9 exactly once.

All three constraints must be true simultaneously.

Below are two 9×9 squares, but only one of them is a Sudoku Square.

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

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

The square on the left is a sudoku square. The one on the right is not, because of an extra "1" in the second row and second column. The lines in the squares above are drawn to highlight the blocks visually in this description.

Usually, Sudoku is posed as a puzzle, with blanks for the puzzle solver to fill in. **We will not be filling in any blanks, or solving any Sudoku problems.** We are simply concerned with checking the validity of a 9×9 square of integers, according to the definitions above.

Requirements

You will develop a software application in Python that does the following:

- Opens and reads a text document containing 9 rows of 9 integers.
- Determines whether or not the 9 rows of 9 integers is a true Sudoku square (see above)
- Reports a simple `yes` if it is, and `no` if it is not.

You can find several example files on the Moodle page for the assignment; the examples named `sudoku_1.txt` etc. will be true sudoku squares, and the ones named `pseudoku_1.txt` etc. will not be true sudoku squares.

The example files will have 9 rows and nine columns of integers. Here's an example:

```
UNIX% more sudoku_3.txt
6 1 9 3 2 8 7 4 5
3 8 7 4 5 6 9 2 1
2 4 5 7 9 1 8 6 3
8 2 1 9 6 4 3 5 7
5 3 6 2 1 7 4 8 9
9 7 4 5 8 3 6 1 2
4 5 2 6 3 9 1 7 8
1 6 3 8 7 2 5 9 4
7 9 8 1 4 5 2 3 6
```

Notice that there are no lines in the data; just a 9 rows of 9 integers each.

What to Hand In

- A text document named `a2q1_design.txt` containing your design plan. Suggestion: your design plan should include the interface of your functions without implementation and the equivalence class and/or boundary cases for your test script.
- A Python program named `a2q1.py` containing your implementation ~~plan~~.
- A Python script named `a2q1_testing.py` containing your test script.

Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

Evaluation

- 10 marks: Your design plan demonstrated careful planning, including functions, testing, and other important aspects.
  - Your design plan document describes a number of functions in terms of inputs, outputs, and purpose. It describes test cases for each function.
- 10 marks: Your implementation meets the requirements.
  - Your application reads the named file containing 9 rows with 9 columns of integers.
  - Your application determines if a 9×9 square is a true sudoku square.
  - Your application outputs `yes` or `no` only.
- 10 marks: Your test script demonstrates careful testing.
  - You have testing for each function in your implementation.
  - Your testing is thorough, and could identify errors in your implementation.

Question 2 (10 points):

Purpose: To reflect on your experience planning and implementing an application.

Degree of Difficulty: Easy

In this question you will reflect on your experience in Question 1. Answer the following questions about your experience designing and implementing the application. You may use point form, and informal language. Be brief. These are reflection questions, and there is no right answer, and there is no need to go back to Question 1 and change anything as a result of these questions. The intent of these questions is to get you to think about the development process and the way you used them to complete your work.

1. (2 marks) Comment on your **development strategy**. For example, you could address the following issues:
   - In your development, did you follow the incremental model or did you think about your tests first so that your functions could be better tested in your unit tests (test-driven strategy)?
2. (2 marks) Comment on your **design plan**. For example, you could address the following issues:
   - How much time did you spend on your design plan?
   - Do you think you planned your application well enough?
   - Did problems arise that you did not plan for? What kinds of problems (if any)?
   - Were there functions you created that were not in your initial plan?
3. (2 marks) Comment on the **implementation stage**. For example, you could address the following issues:
   - Did you estimate the time you'd need to implement the application?
   - Did the implementation take longer or shorter than you planned for, or expected?
   - What took more time than you thought?
   - What took less time than you thought?
4. (2 marks) Comment on your **testing**. For example, you could address the following issues:
   - Did your test script find errors in your functions?
   - Did you discover errors in your functions during your verification stage (when you were running the completed application)?
   - How long did you spend testing and debugging? If you spent more time than you expected, is there any way you can try to reduce the time?
5. (2 marks) Comment on **your use of time**. For example, you could address the following issues:
   - How much time total did you spend, from start to finish (excluding these reflection questions)?
   - Was that more or less than you expected? More or less than you planned?

Remember that the purpose of these questions is to help you learn from your experience.

What to Hand In

Your answers to the above questions in a text file called `a2_reflections.txt` (PDF, rtf, docx or doc are acceptable).

Be sure to include your name, NSID, student number, course number and laboratory section at the top of all documents.

Evaluation

Each answer is worth 2 marks. Full marks will be given for any answer that demonstrates thoughtful reflection. Grammar and spelling won't be graded, but practice your professional-level writing skills anyway.