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 – Solutions

Date Due: January 26, 2017, 10pm **Total Marks: 40**

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

Note:

Before starting Question 1, please read Question 2.

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:

- Runs on the UNIX command line, and accepts one command-line argument, the name of a text document containing 9 rows of 9 integers.
- Opens and reads the text document.
- 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.

Examples

The following is an example of the application's behaviour:

```
UNIX$ python3.6 a2q1.py sudoku_1.txt
yes
UNIX$ python3.6 a2q1.py pseudoku_1.txt
no
```

You can find several example files on the Moodle page for the assignment; the examples named `sudoku_1.txt` etc. will 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.
- 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 runs on the command-line, and uses a command line argument to indicate which file to read.
  - 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.

Solution: In the Solutions folder, you will find

- `a2q1_design.txt` A design document for this question. Functions are described in terms of inputs, outputs, and purpose. Every function has test cases listed, and the functions have very rough pseudocode algorithms.
- `a2q1.py` An implementation of the design document. Functions have doc-strings giving the interface, and are lightly commented.
- `a2q1_testing.py` A test script containing test cases and scripts.

The design features just a couple of functions. This problem did not require a lot. But notice how there is one function that gets called a lot: that makes the design easy to test!

The implementation follows the design closely. I decided to use a list of lists, but it is completely acceptable to use numpy arrays for this. I decided to convert to integers, but that's not strictly necessary; sudoku is not about numbers; it's about logic, and strings are as good here as anything.

The purpose/input/output format was changed to the Purpose/Pre/Post/Return format. This is not necessary; either is fine for this assignment. The commenting is light everywhere else. A few more comments are fine, but too many comments are annoying and unprofessional. Note especially the if-statement that determines whether or not a command-line argument was given. If not, the program does nothing!

The testing of `check()` was quite thorough. I added more test cases that I had planned, because I had a bug in one of my loops. Oops. Testing the `check_all()` function was lighter. I used the CMPT 141 style of testing, which is acceptable, but not strictly necessary. For future assignments, the list-of-test-cases is going to be required, but not for A2.

Many students had a function that took the name of a file as an argument, and then returned a list or array (the data in the named file). This kind of function is hard to test, and I suggested to many that running a few examples at the command-line was enough.

Notes for markers:

- Design document.
  - There was no standard format given, but the solutions for A1 had an example. Students could imitate that. Some students may use input/output etc, and others may use the Purpose/Pre/Post/Return format. Both are fine.
  - Full marks given if each function has purpose/input/output/test cases. A little pseudocode is good but not required.
  - If any one of the purpose/input/output/test cases is consistently missing, 8 marks.
  - If more than one of the items is missing consistently 6/10
  - Give 3/10 for documents that are clearly just not very good.
- Implementation:
  - If the program looks like it meets all the requirements, then give full marks.
  - The program must use `sys.argv` to run on the command line.
  - The program must read a file.
  - The program must do no console output except `'yes'` or `'no'`.
  - The program must check the data in the file.

- Test script:
  - For full marks, 2 to 4 test cases for most functions.
  - Functions that read a file and return a list or array may be tested lightly or not at all. Many students were explicitly told this!
  - If the testing is not in the format of the solution, that's probably okay.
  - A test script that just sends data to the console to allow a human to do a visual check is inadequate. Tests should check values and report if there's an error.

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 waterfall model, or the incremental model?
   - If neither, which was closer to what you did?
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.

Solution: There are no right or wrong answers. There's no way to verify any of the students' comments. Marks here can be generous.

If a student's answer is not consistent with the documents submitted for Q1, you can deduct marks. For example, students who submit answers to this question but not Q1 might not get full marks, depending on what they say.

Notes for markers:

- (2 marks) **Development strategy**.
  - An answer might talk about waterfall, or incremental, or test-driven.
- (2 marks) **Design plan**.
  - Any response that discusses how their planning affected their work is acceptable.
- (2 marks) **Implementation stage**.
  - Any response that discusses their implementation effort is acceptable.
- (2 marks) **Testing**. For example, you could address the following issues:
  - Any response that discusses their testing/debugging effort is acceptable.
- (2 marks) **Your use of time**.
  - Any response that discusses the amount of time they spent is acceptable.