SWEN 6301: Assignment #4

Revision 1

Due on December 21, 2020 at 2:00 PM

50 Points (15% Overall)

Problem 1

(15 points)

Write a high-quality program in your preferred language to output all possible permutations of a given input string. Your program **must not** use recursion, it **must** implement an iterative approach to find all permutations of the string. You are not allowed to use any special string libraries as you are allowed to only use default string operations such as concatenation and substring operations.

1. (*10 points*) Your program should follow the high-quality code concepts and practices we have discussed in class including (but not limited to): code readability and documentation, low coupling and high cohesion, naming convention, code style and organization, and error handling. When documenting your code, please provide meaningful and insightful comments to tell what is the operation performed by a given statement and the intuition behind that operation. Of course, you only need to comment on code snippets that are either ambiguous or need extra reasoning from the reader.

2. (*5 points*) Discuss how your program aligns with the following high-quality coding concepts: code readability and documentation, low coupling and high cohesion, naming convention, code style and organization, and error handling.

Problem 2

(25 points)

The Apache POI project[1] provides pure Java libraries for reading and writing files in Microsoft Office formats, such as Word, PowerPoint and Excel. The source code of the project can be found on Github[2]. Specifically, we are interested in the class `XSSFWorkbook`[3]. This class presents a case of poorly written code with severe problems in coupling, cohesion, style, performance, and debugging. It also lacks proper documentation and user comments. Familiarize yourself with the source code for the class `XSSFWorkbook` at the provided link and answer the questions below:

1. (*10 points*) Write a high-quality version of function `cloneSheet`[4]. Your enhanced version **must** split the function into smaller highly-cohesive functions and should follow the high-quality code concepts and practices we have discussed so far in terms of: code readability, documentation, low coupling, high cohesion, naming convention, code style and organization, and error handling. To get the full credit, you also need to specify your re-factoring strategies that you adopted to achieve the high-quality version.

2. (*9 points*) Assume that the authors for class `XSSFWorkbook` has consulted you to perform a code review. For each of the following cases, briefly discuss your recommendation and the reasoning behind it:

   (a) (*3 points*) Improving the quality of the code between the lines 368−385 in function `onDocumentRead`[5]?

[1] `https://poi.apache.org/`

[2] `https://github.com/apache/poi`

[3] `https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java`

[4] `https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java#L602`

[5] `https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java#L360`

Problem 2 continued on next page...

(b) (*3 points*) The sufficiency of the exceptions and error handling in function `setSheetName`[6].

(c) (*3 points*) The quality (acceptability) of the corrective operation on line 1615[7] in function `setSheetName`.

3. (*6 points*) Determine how many test cases required for function `containsSheet`[8] (lines $1823 - 1842$) using *basis path testing*[9] technique. To receive full credit, show your full work to deduce the answer and write down at least 2 possible unit test cases.

Problem 3

(10 points)
The `SensorDataProcessor` class[10] is an example of inefficient and poorly written code. Study the code and answer the following questions:

1. (*6 marks*) Suggest at least 3 strategies to speed-up the code between the lines $36 - 52$ in function `calculate`. Use line numbers to communicate your strategies.

2. (*4 marks*) Perform code jamming for the code between the lines $36 - 58$ in function `calculate`.

[6] https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java#L1605

[7] https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java#L1615

[8] https://github.com/apache/poi/blob/REL_4_1_1/src/ooxml/java/org/apache/poi/xssf/usermodel/XSSFWorkbook.java#L1823

[9] https://en.wikipedia.org/wiki/Basis_path_testing

[10] https://gist.github.com/atamrawi/a0f0f79b76d04618ff0420eb19b97dac