NATIONAL UNIVERSITY OF SINGAPORE

CS2103/T – SOFTWARE ENGINEERING

(Semester 1: AY2015/2016)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number only. Do not write your name.
2. This assessment paper contains **SIX** questions and comprises **TEN** printed pages.
3. Answer **ALL** questions within the space in this booklet.
4. This is an OPEN BOOK assessment.

STUDENT NO: _____

This portion is for examiner's use only

Question	Marks	Remarks
Q1	/6	
Q2	/6	
Q3	/5	
Q4	/10	
Q5	/6	
Q6	/7	
Total	/40	

The **problem description** used for all questions in this assessment paper is given in the appendices (last two pages of this assessment paper).

Q1 [2+2+2=6 marks]:
Please read *Appendix A* of the problem description. When answering this question, you may consider the requirements specified in the problem description as well as requirements that you would like to add.

a) Assume that you have categorized CollatePlus user stories into categories *must-have*, *nice-to-have*, and *unlikely-to-have*. Give one *must-have* and two *nice-to-have* user stories for CollatePlus. All three user stories must help the user evaluate the code quality of the code written by a student.

1. [must-have]

2. [nice-to-have]

3. [nice-to-have]

b) Give an example of a *non-functional requirement* of CollatePlus that is directly related to a functional requirement specified in your answer to part (a) above.

c) Give two *extensions* to the use case description given in *Appendix B*. Write your answer below (not in *Appendix B*).

Q2 [4+2=6 marks]:

(a) Propose an architecture for CollatePlus. Your architecture must include the components given in *Appendix C*. If your architecture contains more components (recommended), please provide a brief overview of each component you added (similar to the descriptions given in *Appendix C*).

(b) Consider this extract (only two components are shown) from another proposed architecture for CollatePlus. Comment on how the two arrows between GUI and Logic affect the design quality.
Is it possible to remove one of them?
If yes, which one and how? If no, why?

Q3[5 marks] Consider the use case description given in *Appendix B*.
Use a suitable UML diagram to explain the interactions between architectural components required during step 5-6 only.
State (in the diagram) the API methods taking part in the interactions.

Q4 [7+3=10 marks] (a) Use suitable UML diagrams to propose an object-oriented design for the `Data` component. Show all *navigabilities* and *multiplicities*. Try to minimize *associations*.
Show important *attributes* and important *methods* only (no more than 5 each).
Your design must include, but is not limited to, the classes listed in *Appendix D*.

(b) Consider the sample data given in *Appendix E* and your proposed design above. Use a suitable UML diagram to show the object structure in the `Data` component for the given sample data.

Q5 [4+2=6 marks]:

(a) Consider the getStudents method/function in *Appendix F*. Use Java, C++, or pseudo code to show how you would use i. *assertions*, ii. *exceptions*, and iii. *logging* in that method.
Show no more than one example of each. Only the relevant lines of the method need to be shown.

(b) If the Analyzer component consists of just one class and it is instantiated no more than once at any time, is it OK to make all its members class-level (i.e. static)? Justify your answer.

Q6 [5+2=7 marks] (a) Assume you have been asked to test the getStudents method described in *Appendix F*. Propose an effective and efficient set of tests cases. You may not propose more than 8 test cases.

Given below is an example test case list used for testing a different method isCorrect(String answer) from a different software. You may follow a similar format in your answer.

#	answer	Is question open for submission?
1	Correct answer	Yes
2	Incorrect answer	No

(b) Comment on the following statement.

While 100% *path coverage* is very difficult to achieve, it is worthy of achieving because it certifies the code as bug free.

[This page may be used if you need extra space for any of the answers]

--- End of Paper---

Given below is the **problem description** used for all questions in this assessment paper. You may detach this page from the assessment paper. There is no need to submit this page.

Appendix A. CollatePlus product description

Your team has been asked to build a software application called **CollatePlus** which will be used for analyzing text (code and documentation) written by students in CS2103 module project.

1. CollatePlus is a desktop application. It is meant to be used by CS2103 teaching team members.
2. CollatePlus can be accessed using a GUI or a Textual UI (TUI).
3. CollatePlus downloads student data from the IVLE server and commit history from the GitHub server. All data for the entire cohort is downloaded at once. The download is triggered by the user. Data from GitHub comes as a single string. Data from IVLE too comes as a single string. The data is then analyzed and converted into an object structure that captures the information required to display data about student contributions to the project.
4. For any student in the class, CollatePlus can show,
    a. A collation of all the text segments (code and documentation) written by the student for the project.
    b. A list of all the commits made by the student in the team repo.
    c. A list of all the project files modified by the student.
5. CollatePlus does not store the downloaded data in the user's Computer.
6. While multiple students may have modified a text segment, only one student may claim the authorship of a text segment. The author of a text segment is indicated using specially-formatted comments, similar to how you did it in your module project.
   Some segments may not be claimed by any student (e.g. code reused from elsewhere).
7. Some configuration data (e.g. login credentials of the user for IVLE/GitHub, mapping from matric number to github user ID, etc.) are to be put in a file named config.txt. The user is expected to create that file manually, following a specific format. CollatePlus reads that file at startup.

Appendix B. An example use case description

Use case: View text written by a student
Actors: Lecturer

1. Lecturer launches the Text UI version of the app.
2. App prompts the Lecturer to enter a command.
3. Lecturer enters the command to download data from servers.
4. CollatePlus downloads data from GitHub and IVLE.
5. Lecturer enters the command to analyze data.
6. CollatePlus analyzes the downloaded data and prepares the internal object structure.
7. Lecturer enters the command to view text written by a specific student.
8. CollatePlus shows a listing of all text written by the specified student.

Use case ends.

Appendix C. Suggested components for the architecture

- **GUI**: The Graphical User Interface.
- **Logic**: The main logic of the application.
- **Analyzer**: A temporary component used only during analysis of the downloaded data. The component is created at the beginning of analysis and discarded after the analysis.
- **Data**: Created by the Analyzer component. Holds the object structure containing the data required to answer user queries.

Appendix D. Suggested classes for the Data component

- Segment: An object of this class represents a segment of contiguous text authored by a particular student. E.g. if a student is claiming authorship of all text in a particular file, all text in that file is considered one Segment.
- File: Represents a file in a particular team's repo. This class supports a compact() method that removes unnecessary spaces from the file. What is considered 'unnecessary spaces' depends on whether the file is a documentation file, java file, c++ file, xml file, etc.
- Commit: Represents a commit made by a student. (note: CollatePlus only keeps track of who made the commit, when the commit was made, and which text segments were affected)

Appendix E. Sample data

- Adam and Betsy are in Team1.
- Adam has made only one commit. In that commit he added the file readme.txt with some content in it.
- Betsy has made only one commit. In that commit she added some text to the end of readme.txt and modified some of the Adam's text in it.
- Adam is claiming authorship of text segment he added. Betsy is claiming authorship of the segment she added.

Appendix F. getStudents method

getStudents (String moduleCode)

Connects to the IVLE server and retrieves a string containing details of students in the specified module if the logged in user is a teaching team member of that module. It is assumed that the connection to the IVLE has been set up before calling this method.

-- End of problem description--