COSC 442

Software Quality Assurance & Testing

Spring 2017

**Exam II – Part B**

**Name: / 30**

This portion of the exam consists of **2** questions and is open book/computer. You may elect to use any quality assurance / testing technique, tool and/or coverage criteria we discussed in class or that you used your projects. You must answer **all** questions. Please read through all questions prior to beginning the exam. At the completion of this portion of the exam, please submit this document, with your answers, to Blackboard.

**Grading:**

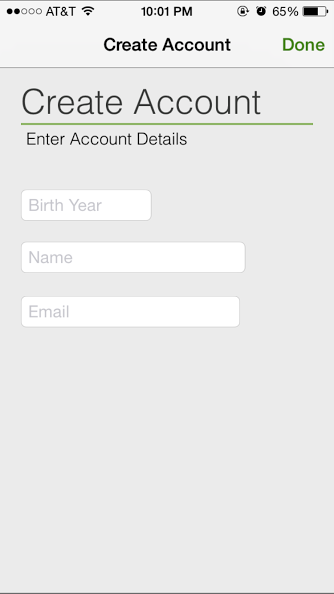
Your scores on these questions will depend on the extent to which:

* The answer is complete.
* The answer demonstrates exemplary knowledge of the question.
* The answer is clearly organized and easy for the reader to follow.
* The answer is free of grammatical and spelling errors.

*Whoever ceases to be a student has never been a student.*

~ George Iles

1. (10 points) The [Wavyleaf Android](https://play.google.com/store/apps/details?id=com.towson.wavyleaf) mobile application (account creation screen shown below) was developed at Towson University as a citizen-science application for users to report the spread of [wavyleaf basketgrass](http://www.dnr.state.md.us/wildlife/Plants_Wildlife/WLBG/index.asp) in Maryland.



We have developed a number of automated functional (i.e., acceptance) tests using the capture-and-replay tool, Robotium. The test suite takes approximately 15 minutes to run against an Android device (assume no setup time). This application needs to be functional for all the Android configurations defined at <http://developer.android.com/reference/android/content/res/Configuration.html>; however, there are more than 170,000 total configurations, requiring more than 24 staff-years of testing. To make the testing feasible, we reduced the configurations we are interested in only the following configurations:

Parameter Name Values

HARDKEYBOARDHIDDEN NO, UNDEFINED, YES 3

KEYBOARDHIDDEN NO, UNDEFINED, YES 3

KEYBOARD 12KEY, NOKEYS, QW ERTY, UNDEFINED 4

NAVIGATIONHIDDEN NO, UNDEFINED, YES 3

ORIENTATION LANDSCAPE, PORTRAIT, SQUARE, UNDEFINED 4

TOUCHSCREEN FINGER, NOTOUCH, STYLUS, UNDEFINED 4

If you only have 40 hours with which you can run your test suite against the configurations, at what strength (i.e., 2-way, 3-way, 4-way) of combinations would you recommend to test against? Justify your answer and provide the test configurations.

1. (20 points) The time allotted for software quality assurance and testing is never enough. Thus, you will often be faced with making the biggest software quality assurance impact you can in the time prior to a deadline.

We started the semester with examining the [MegaMek](http://megamek.info/) game, a modern turn-based strategy game; we will end the semester with examining a simplified version of [Nethack](https://en.wikipedia.org/wiki/NetHack) (i.e., JoshHack), a modified[[1]](#footnote-1) rougelike computer game originally released in 1987 with old-school ASCII graphics, posted as an archived Eclipse project to Blackboard under Exams/Exam II. In this game, your goal is to “descend the Caves Of Slight Danger, find the lost Teddy Bear, and return to the surface to win. Use what you find to avoid dying.” The game can be run from Eclipse as a Java Application in the standard way. Once presented with the main screen (shown below), you can press the *?* key for help and playing instructions.



While the game is functional (and fun to play!), we are interested in improving quality of the software as a whole. For this question, your task is to simply improve the software quality of this game as you see fit. You may use any/all of the tools we’ve talked about throughout the semester and/or any of the tools you used for the final project.

Throughout this question, you should commit this project and any incremental software quality improvements you have made to a new repository (named cosc442-exam2) in your individual GitHub repository (with appropriate commit comments!) and add COSC442Spring2017 as a collaborator. To ease things, you can simply commit the individual classes that you improved (not the entire project) using the GitHub.com drag and drop method of committing. Finally, provide completed table (see below) that describes/justifies what you did to improve the software quality. I have provided two examples entry as guidance.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Issue** | **Improvement** | **Tool(s)** | **Rationale** |
| *FieldofView.java* | *Cyclomatic complexity is too damn high!* | *Refactored the method to a complexity of 5 by extracting several helper methods.* | *JDeodorant, CodePro* | *Reducing the cyclomatic complexity will allow for easier software maintenance and code comprehension, which may uncover existing faults. Also promotes better design principles.* |
| *Inventory.java* | *No test cases / 0% coverage* | *Developed/generated unit tests to achieve 80% code coverage.* | *JUnit, EclEmma.* | *Providing unit tests enables regression testing and helps identify potential software faults.* |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Modified from <http://trystans.blogspot.com/> [↑](#footnote-ref-1)