CS408: Incremental Testing and Regression Testing Grading Rubric

Do Incremental Testing and Regression Testing on your software product using the test cases from your Test Plan. After each of the two steps, fix all defects found. Create a log (report) of all defects found by Incremental Testing and Regression Testing and how they were corrected. Also, update the status of your requirements in the product backlog.

1 Classification of Components (15 points)

1.1 Define all components - 10 points

- We expect a diagram explaining various components in the project. You should outline the base components and other independent components, and show the interaction among the components - 5 points
- Specify the input, output and the dependent components (components it calls and the components that call it) for each component - 5 points

1.2 Which form of incremental testing did you follow - 5 points

- Top-down or Bottom-up? Explain.

2 Incremental and Regression Testing (80 points)

2.1 Automation - 10 points

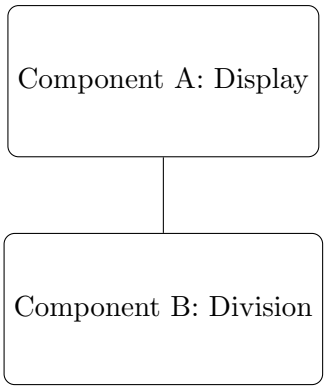
Specify whether you automated the incremental and/or regression testing. If yes, give a brief description of your automation framework/strategy.

2.2 Defect log - 70 points

- Incremental and regression testing should be conducted on a per module basis. We will be looking for testing to be completed on each of your modules.

We expect you to list down the defects found during incremental testing and regression testing of each module. The defects will be evaluated based on how well the defect is described, the severity assignment and the solutions provided.

- Given below is a sample regression test case example for a calculator:



The main component is the display component (Component A), which should have test cases for displaying digits. Component B is a division component, which depends on Component A. The test cases for Component B should verify if a division is performed correctly and look at boundary values (division by zero). The regression test case for the above example should verify if division by zero works in Component B, that it also works properly in Component A (i.e. the existing functionality is not broken after a defect is fixed).

- Given below is a defect log sample for the above example:

Module	Component B - Division

Incremental Testing

Defect No.	**Description**	**Severity**	**How To Correct**
1	Application should not crash when dividing by zero	1	Create an invalid divide by zero exception
2			

Regression Testing

Defect No.	**Description**	**Severity**	**How To Correct**
1	Fixing division by zero in component B (Division) causes component A (Display) not to display	1	Catch exception created by Division module and print ERROR on display
2			

3 Update Product Backlog (5 points)

Update the status of your requirements:

- If a requirement planned in the current sprint is complete, change its status to 'Completed in sprint <number>'
- If a requirement planned in the current sprint is incomplete, change its status to 'Incomplete: moved to sprint <number>'
- If a requirement planned in the current sprint is partially complete, change its status to 'In-progress: moved to sprint <number>'
- If a (not yet planned) requirement is selected for the next sprint, mark its status as 'Planned for sprint <number>'