UNIVERSITY OF LONDON

BSc in Computer Science

Module: CM2010 Software Design and development

Midterm coursework

This coursework consists of 3 parts, and you should answer all parts.

Part 1: Module coupling and cohesion

For this part of the coursework, you are to analyse a program in terms of module coupling and cohesion. The output of this part is a short report containing extracts of source code along with descriptive writing.

Instructions

You can choose to analyse any program that you personally have worked on and that you have access to the source code. You might choose a program you have seen during your studies on the BSc CS degree or another program.

Carry out the following tasks:

* Examine the code and identify two different types of module coupling that are happening in the code. In your report, put in extracts of the code which illustrate the coupling and explain why you think that type of coupling is happening there.

* Examine the code and identify two different types of module cohesion that are happening in the code. In your report, put in extracts of the code which illustrate the cohesion and explain why you think that type of coupling is happening there.

Here is a checklist for part 1:

	Done	Marks
Clearly state which program you are analysing and how you found that program		N/A
Module coupling example 1: code extract and explanation		2.5
Module coupling example 2: code extract and explanation		2.5
Module cohesion example 1: code extract and explanation		2.5
Module cohesion example 2: code extract and explanation		2.5
Total		10

Note 1: Please ensure that you type the report and submit it in PDF format. Failure to comply with this requirement may lead to a score of 0 for this question.

Note 2: The word limit for this task is approximately 500 words. Exceeding this limit by more than 10 percent will result in a deduction of 2 marks.

Part 2: Unit testing activity

In this part of the coursework, you are to demonstrate your ability to carry out test-driven development according to the three rules seen in the course. You can work in C++, Python or JavaScript for this part.

The scenario is that you are working on a space battl-themed video game, and you must write some of the collision detection and damage computation code. First, you should develop the following function:

```
point_is_in_box(pointX, pointY, boxTopCornerX, boxTopCornerY, boxWidth,
boxHeight)
```

It should return true if the sent pointX, Y falls inside the box specified with boxTopCornerX, boxTopCornerY, boxWidth, boxHeight.

Develop this function using three cycles of test cases. Gather evidence of your test-driven development process – you will need to present the evidence in your report:

- Test 1:
  - Simple test code that the function cannot pass
  - V1: Updated code for function that passes that test
  - Console output showing it initially failing the test then passing it
- Test 2:
  - Simple test code that the function cannot pass
  - V2: Updated code for function that passes that test
  - Console output showing it initially failing the test then passing it
- Test 3:
  - Simple test code that the function cannot pass
  - V3: Updated code for function that passes that test
  - Console output showing it initially failing the test then passing it

Now repeat the three cycles for another function:

```
compute_damage(weapon_power, playerX, playerY, explosionX, explosionY)
```

This function should return a damage value. It should use some method to increase the value of the returned damage value according to the player's spaceship and the explosion position – closer to the explosion means higher damage, higher weapon power means higher damage.

Again, run the test-driven development cycle three times on this function, gathering evidence as above.

Finally, write everything up in a report which you should save out to PDF format. In the report, you should present the test code, the test strategy (why this test?), the initial function code that fails the test, then the function code that passes the test. Present test output, different versions of your functions and tests as clearly as possible using appropriate formatting. Do not simply copy paste screen grabs – we want to see properly formatted text in your report. Complete your report with a short reflection on your experiences with test driven development, identifying positive and negative aspects.

Here is a checklist you can use to ensure you have completed all the tasks:

	Done?	Marks
point_is_in_box function: 3 test cycles, 3 lots of test-fail-test-pass evidence		3
compute_damage function: 3 test cycles, 3 lots of test-fail-test-pass evidence		3
Reflect on the process of test-driven development identifying positive and negative aspects		2
Report in PDF format with properly formatted code and console output. Aim for around 500 words PDF as a report.		2
Total		10

Part 3: Secure programming

For this part, you are to develop a strategy to improve the security of a piece of software using secure programming techniques. The output of this part is a report in PDF format.

Instructions

Assume you are working on a web application – like web applications you saw in this course. The application includes a login page to check a username and password using a database. Additionally, the application provides a separate registration page, allowing new users to sign up, to define a username, and to set a password. This application stores these new usernames and passwords into the database. The client sends everything as a text (without using any encryption algorithm) to the server using the GET method. Moreover, the passwords are stored in the database as plain text. The other parts of the application can be anything.

Referring to the strategies in "OWASP's secure coding guidelines" available at https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/assets/docs/OWASP_SCP_Quick_Reference_Guide_v21.pdf, describe FIVE things you can do to the code in that application to improve its security. Report your secure programming plan. Note that you do not need to actually implement the secure code, just report on what you plan to do.

Here is a checklist for part 3.

Items	Done?	Marks

Secure programming recommendation 1: what is the problem? What should change and how?		2
Secure programming recommendation 2: what is the problem? What should change and how?		2
Secure programming recommendation 3: what is the problem? What should change and how?		2
Secure programming recommendation 4: what is the problem? What should change and how?		2
Secure programming recommendation 5: what is the problem? What should change and how?		2
Report in PDF format containing secure programming recommendations, as described above. Aim for around 500 words.		If it is not, a penalty would be considered (-2 marks)
Total		10

What to submit

You should submit the following:

Part 1: PDF report, approx. 500 words.

Part 2: PDF report, approx. 500 words.

Part 3: PDF report, approx. 500 words.