Assignment Specification -V1

Learning objectives

1. Design test cases using black-box and white box-testing approaches.
2. Implementing a test code and test a production code.
3. Design and develop code adhering to basic modularity concepts and verify whether code is following basic modularity concepts and refactor if not.
4. Discuss ethical and professionalism issues in Software Engineering projects.
5. Use version control and keep track of a small Software Engineering project.

1. Introduction

You have learned about basic concepts of software testing, modularity, version control and ethics in the second half of this unit. You learned how to use a version control system to maintain your code and other files. You have learned how to design test cases using black-box and white-box testing approaches, implement test designs as a test code and run them to test your code. You have learned about several concepts related to designing better, more modular code. You were introduced to ethical and professional issues in software engineering and ways to reduce them.

This assessment evaluates your competency in above areas of Software Engineering. It expects you to apply the knowledge you have gained via lectures, practical classes and sign-offs to design and implement simple software following modularity principles, verify your code against a review checklist, design test cases and test the code you have developed. Finally, you will explore ethical and professionalism issues and ways to avoid them in your software system. You will use a version control system to keep track of all the activities of the whole assessment task. You will document your work in each stage so that another person can understand the work you have done.

2. Scenario

Assume you are involved in a project which plans to produce software which can perform different forms of conversions. String conversions and conversions of measurements such mass, length and time are some areas you are going to design and implement. Additional functionalities relating to strings and measuring units will also be implemented.

Some functionalities expected are:

Category 1

a. Converting a given string to upper case or lower case.
b. Identify whether numeric values are in a given string.
c. Identify whether a given string is a valid number or not.
d. Remove any numeric values in a given string and then convert the string to upper case or lower Case.

Category 2

a. Converting a number which represents a length given in meters to feet and vice versa and centimeter to inches and vice versa.
b. Converting a number which represents a mass given in kilograms to pounds and vice versa, and mass given in milligram to ounce and vice versa.
c. Converting a number which represents a time given in hours to minutes and vice versa, and time given in minutes to seconds and vice versa.

Choose how to take input and output:

For all functionalities, inputs/outputs can be handled differently. In the basic version, string/ number can be a parameter and the result can be returned. Additional functionality such as allowing a user to input a string via keyboard and printing the results on the screen and is also to be considered. Finally, functionality to read input data from a txt file and to save the results to a text file is expected to be incorporated. For Category 2, input/ output text file would be a set of numbers in a file, either separated by a space or in separated lines (you can consider either of them).

3. Detailed description

Considering the given scenario, you will design and implement a production code following modularity guidelines, design test cases and test your production code using proper testing processes and analyze the ethical considerations of your code. You will use basic version control methods to keep track of your work.

You are expected to consider all the functionality of Category 1 and only one of (a) (b) (c) of Category 2.

Your work will be organized as seven parts as described below.

1. Version Control
   Apply version control process to keep track of a simple software project [15 marks]

   Read all the assessment tasks and get an idea of what you are supposed to do in this assignment. You are supposed to use version controlling and keep track of all your work (documents and code both) relate to this assignment in a Git repository. Derive a short plan, identifying what branches will you need and why and when the branches will be merged.

   Now create a Git local repository for this assessment using your surname and ID as part of the repository name following the format <surname>_<firstname><ID>_ISErepo). Then, commit all code and documents you create in the rest of the assessment ***as you create and modify them*.** Make sure you commit your changes as and when you are doing them, but not all at once at the end.

   All your code must be in a directory called “code” and documents should be in a directory called “documents”.

   Note: There is no hard rule about what each commit should contain, and you are expected to show your ability to use version control meaningfully.

2. Preliminary description of your modules: [5 marks]

   Looking at the given scenario, write down description/s for modules you decided to implement. Apply the modularity guidelines you have learned in the lectures/ worksheets when identifying and planning modules. Descriptions should include short name, clear and detailed explanation on what the intended task of the modules and how it behaves and, inputs and outputs.
   Indicate whether inputs are taken from keyboard, files or passed to modules as parameters and how outputs are made available (such as a return values, print on screen etc.). You should decide on suitable, meaningful names for the function/s and variables.

   In this unit, "modules" refers to any sub-part of a program.
   The module/s you design in this stage affect the rest of your assessment. Therefore, read the whole assignment properly before deciding on what tasks you are going to do. You may use an iterative approach especially for task 2 and 3.

3. Modularity [20 Marks]

   Implementation of the production code

   a. Implement the modules designed in the part 2 above considering the good modularity principles discussed in lectures/ worksheets.
      You can use either Python (Python3) or Java for your implementation. Execute and verify that you have got your code running without syntax errors.
      Note: This step will create a production code and not the test code.
   b. Create a short review checklist to check whether you have followed good code design principles. You are expected to cover all basic guidelines covered in the Modularity lecture.
   c. Review your code using the checklist, identifying any issues.
   d. If you have identified any issues, refactor your code to address such issues.
   e. After refactoring, if your modules are changed, revise your preliminary descriptions of your modules as revised module descriptions.

4. Test designs (Black box testing) [12 Marks]

   Based on the module descriptions written by you in part 2 above (OR the revised version after refactoring in Part 3 above, if any revision is done), design suitable test cases using:
   a. equivalence partitioning.
   b. boundary value analysis.

   You may design test cases using both above approaches for all the modules you have defined OR you can use Equivalence partitioning for some modules and Boundary value analysis for other modules. Mention clearly which approach is used in each test design.

   Your test data must include all following data among other test data items:

   Last four digits of your student ID, your last name, your full name as appear in your student ID card, name of a movie you wish to watch

5. Test design (white-box testing) [6 marks]

Look at your code implemented above and identify at least two modules where white-box testing approach will be beneficial. Design test cases to cover functionality of the selected modules using white-box testing approach. Indicate clearly which modules are tested with white-box approach.

6. Test Implementation [20 Marks]

Implement your test designs for part 4 and part 5 above using either Python or Java. Using a unit test framework is optional.

Run your test case and obtain results. Identify any test failures and then try to improve your code.

7. Ethics and Professionalism [15 Marks]

Think about a scenario the code you have designed can be used in a large software project.

a. Discuss how lack of ethical and professionalism can result in harmful effects using the code you have designed and implemented. You may refer to lecture 9, seven points useful in identifying ethical issues, as a guidance.
b. Using ACS or IEEE-CS Ethical guidelines, give two suggestions to avoid ethical and professional issues in your software proposed in this assignment.

- You may note the total marks as per the above detailed description is 93. The rest of the 7 marks will be allocated for the documentation (Cover page, Introduction, Discussion, format)

4. Documentation

You need to document what you have done in each stage of the assessment so that another person can get a clear idea about what you have done. You are expected to produce a short report, collating all your work as part of your submission.

Your report name must be of the following format:

" < YourFirstName><YourLastName>_<your student ID>_Report"

Your report must be written in markdown. Refer https://www.markdownguide.org/ if you are not familiar with the markdown (.md) format.

Your report should include following sections:

A. Cover page
   Include the assessment name, your name, Curtin student ID, practical class (date/ time / venue)
B. Introduction
   What functionalities you have decided to implement and overview of work you have done.
C. Module descriptions

Original module descriptions you have created for the part 2 of this assessment, Assumptions you made, if any, and any explanation on why you have chosen to implement these modules.
Revised module descriptions resulted after refactoring, if any (after doing the part 3 the detailed description)

D. Modularity

Description on how to run your production code with correct commands.
Discussion on how different modularity concepts are applied in your code, review checklist you have created, results of conducting the review using the review checklist with explanation on your results and how you have addressed any issues.
You can use screen shots to support your answer in this part.

E. Black-box test cases

All test cases you have designed in the part 4 of this assessment, assumptions you made if any, and brief explanation why you have done the test design in the way you have done.

F. White-box test cases

All test cases you have designed as answer for the part 5 of this assessment, assumptions you made if any, brief explanation why you have done the test design in the way you have done

G. Test implementation and execution

Description of how to run your test code with correct commands.
Results of test execution with test success and failures with short discussion of results from part 6 of this assessment, discussion on whether you have attempted to improve your code and new results, if any.
You can use screen shots to support your answer in this part.
You are supposed to produce a table which shows following information to help you and the marker to check the work you have done. (EP: Equivalence partitioning, BVA: Boundary value Analysis, BB : Black-box, WB: White-box)

Module name	BB test design(EP)	BB test design (BVA)	WB test design	EP test code (implemented/ run)	BVA test code (implemented /run)	White-Box testing (implemented/run)
xxxxx	done	done	not done			
yyy	done	not done	done			
….						

H. Version control

Log of the use of your version control system, any explanation /discussion on version control. (refer part 1 of the detailed description)

I. Ethics

Answer to part 7 of this assessment.

J. Discussion

Reflection of your own work, ways to improve your work and any other thing you wish to present.

Your report would be around 12-25 pages.

5. What you will be submitting

Your submission will be done in **two** steps.

Submission step 1:

Your documentation, i.e. your report (refer section 4), should be submitted to the **Turnitin link (Assignment submission: step 1 documents only)** provided in the assignment folder. Your report submitted to this link should be in PDF format. (You must convert your report in markdown format to a PDF file before submitting to this link). Your report name must follow the following format :

" < YourFirstName><YourLastName>_<your student ID>_Report"

Submission step 2 :

You must submit single zip file of all the work produced in this assessment to the "**Assignment submission: step 2**" link provided in the assignment folder. First, create a folder with name

< YourFirstName><YourLastName>_<your student ID>_ISEAssignmnet. Then place all your work inside this folder. Example : *JohnWhite_12134567_ISEAssignmnet*

Avoid using file formats other than `.zip` file to reduce delays in marking.

The folder should contain:

a. All you code files: You have to submit all your code files (production and test codes), any sample data files (if created) and any other file resulted in part 3 and 6 of this assessment. Name your files in appropriate manner.
b. A single Readme file: a short text file indicating the purpose of each file you have submitted.
c. .git sub directory of the Git repository you have used in the assessment.
d. Your report (refer section 4) in markdown format (.md file)
e. Your report (refer section 4) in PDF format which was already submitted to the Turnitin link.
f. A signed and dated assignment cover sheet in PDF format. Assessment coversheet is available under Assessments page of Blackboard. You can sign a hard copy and scan it or you can fill in a soft copy and digitally sign it.

Zip this folder and submit to the "**Assignment submission: step 2**" provided in the assignment page.

Make sure that your zip file contains what is required. Anything not included in your submission may not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct.

6. Marking rubric : How you will be evaluated ?

Marks will be given out of 100 as mentioned in the section 3 and will contributed to 50% of the unit's total assessment marks as specified in the unit outline.

Your work will be assessed based on (a) code submission (b) Report and (c) Demonstration of your work. Look at section 5 of this document for submission instruction. A short demonstration (less than 15 minutes) will be held after the submission and details will be announced later.

Code submission Implementation of production code and test code (Part 3 and Part 6 of the detailed description) and .git directory (version controlling).	20 marks
Demonstration of your work	20 marks
Report (refer section 4)	60 marks

7. Requirement to pass the unit

As specified in the unit outline, you should score at least 40% of the final assessment to pass the unit. This assignment is your final assessment; therefore, you need to get at least 40% of the marks of this assignment to pass the unit.

Marks of the assignment is given out of 100 and the assessment is worth 50% of your final mark (overall mark of the unit).

Exact mark breakdown in Section 3 and Section 6 of this document represents maximums, achieved only if you completely satisfy the requirements of the relevant section.

8. Plagiarism

Plagiarism is a serious offence. This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). Please read the *Coding and Academic Guidelines* on Blackboard and For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au

In summary, this is an assessment task. If you use someone else's work or assistance to help you complete the part of the assessment, where it's intended that you complete it yourself, you will have compromised the assessment. You will not receive any marks for any parts of your submission that are not your original work. In the case of doubt, you may be asked to explain your code and the reason for choices that you have made as part of coding to the unit coordinator. A failure to adequately display knowledge required to have produced the code will most likely result in being formally accused of cheating. Finally, be sure to secure your code. If someone else gets access to your assignment for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

9. Late submissions

You must submit the assignment on the due date. Acceptance of late submissions is not automatic and will require supporting documentation proving that the late submission was due to unexpected factors outside your control.

Note that external pre-scheduled commitments including, but not limited to, work, travel, scheduled medical, sporting, family or community engagements are not considered for unexpected factors outside your control. If you know you have, or are likely to have, such engagements and that they may affect your ability to complete the assignment, you will be expected to have planned your work accordingly. This may mean that you need to start and/or complete your assignment early to make sure that you are able to hand it in on time. Also note that IT related issues are almost never a valid excuse.

In the event that you submit your assignment late and you will be penalised with a with a 5% penalty for 1 day late, then an extra 10% penalty for each calendar day late after that up to a maximum of seven (7) calendar days, as per the university policy. Any work submitted after this time will not be marked and you will automatically fail the unit.

Note that if you are granted an extension, you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

10. Clarifications and Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced in the lecture and on the unit's Blackboard page (not necessarily at the same time and not necessarily in that order). These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these, either by attending the lectures, watching the recordings and/or monitoring the Blackboard page.

11. General instructions

- Remember to start small and come back and revise your work specially in part 2 and 3 of the detailed description. If you spend more time in thinking and designing what you would do, rather than quickly try to implement something, you would be able to do much better in this assignment. Think about the total marks allocation (given as out of 100) , time you would spend on each section and mark allocation for each section very carefully.
- Read the assignment fully before start answering the assignment. The design decisions you will taking in initial sections of the assignment will affect your ability to showcase your competency in testing and modularity. Therefore, spend reasonable time on thinking about what you are going to do and how you are going to do it, before starting coding/ test designs.
- You may need to revise your work few times until you are satisfied with your work. Your version control history will show these revisions.
- Design your test cases to demonstrate wide range of abilities you have developed such as:
    - Testing numeric values
    - Testing strings
    - Testing keyboard input/ console outputs
    - Testing file input/outputs.
    - Testing loops/ if-else statements/ exceptions.
- Start your work by creating a repository and make sure all your work from the beginning is stored there. If you do all your work in your local machine without committing to Git and commit only the final version there, you will not be able to show your ability to use basic functions of git effectively.
- Your submission to Blackboard and your Git repository will be evaluated separately to avoid use of Git affecting other parts of the assignment.
- Your code must run in a Linux command line environment. Your code can be in either Python or Java. Make sure you can demonstrate your work in lab environment.

End of the Assignment Specification