Assessment – Monster Card Catalogue

Achieved	Merit	Excellence
91896: Use advanced programming techniques to develop a computer program (6 credits)		
Use advanced programming techniques to develop a computer program.	Use advanced programming techniques to develop an informed computer program.	Use advanced programming techniques to develop a refined computer program.
91897: Use advanced processes to develop a digital technologies outcome (6 credits)		
Use advanced processes to develop a digital technologies outcome	Use advanced processes to develop an informed digital technologies outcome.	Use advanced processes to develop a refined digital technologies outcome.

Introduction

This assessment activity requires you to plan, trial, test and develop a computer program using advanced programming techniques. You will use a development process to help you make informed decisions throughout the coding, testing and trialling of your program and show ongoing refinement to improve the functionality and quality of your program.

You will be assessed on how effectively you plan your development, decompose the outcome into smaller components, and test and refine your program so that it is a high-quality response to the task (e.g. well-structured, logical, flexible, robust and comprehensively tested).

When planning and developing your program, you must ensure your program:

- uses variables storing at least two types of data (e.g. numeric, text, Boolean)
- uses sequence, selection and iteration control structures
- takes input from a user, sensor(s), or other external source(s)
- produces output

AND includes two or more advanced programming techniques, such as writing code that:

- modifies data stored in collections (e.g., lists, arrays, dictionaries)
- defines and manipulates multidimensional data in collections
- creates methods, functions, or procedures that use parameters and/or return values
- responds to events generated by a graphical user interface (GUI)
- requires non-basic string manipulation
- uses functionality of additional non-core libraries.

Task

Your friend is creating a Monster Card game. As part of the planning for the game, they want to create a catalogue of all the cards that will be included in the game. They have asked you to create a computer program that will create the catalogue.

Specifications

Your program needs to:

- Store the details of the monster cards that already exist in the catalogue. See the table on the next page listing the existing cards. These can be coded as part of the program (e.g., as a dictionary at the start of the program).

  NOTE:

  - There are only four possible traits categories for monsters:

    strength, speed, stealth, and cunning

  - The values for all categories are from 1 to 25. In other words, the minimum strength a monster could have is 1, and the maximum strength they could have is 25

- Allow the user to add new monster cards to the catalogue
  NOTE: Once the user has added a card, the program should display the card that has been added, and check with the user whether the details are correct, or if they want to make a change.
- Allow the user to search the catalogue for an existing monster card, check that the monster's details are correct, and make changes if necessary.
- Delete a monster card from the catalogue
- Output the full menu to the Python console (to allow for it to be printed)
- The computer program needs to use a graphic user interface (GUI) to get input from the user.

Any changes made to the catalogue only need to apply while the program is running. In other words, once you exit the program, changes to the menu do not need to be saved.

Monster Card Catalogue: Existing Cards

Name	Strength	Speed	Stealth	Cunning
Stoneling	7	1	25	15
Vexscream	1	6	21	19
Dawnmirage	5	15	18	22
Blazegolem	15	20	23	6
Websnake	7	15	10	5
Moldvine	21	18	14	5
Vortexwing	19	13	19	2
Rotthing	16	7	4	12
Froststep	14	14	17	4
Wispghoul	17	19	3	2

What you need to do – follow these steps:

1. Decide on an appropriate plan, and then set up the project management and version control tools you will use to manage your program development.
2. Set up your evidence recording document e.g., the PowerPoint template provided.
3. Decide how you will collect input from your users and how you will structure your output:
    a. How will you get users to give the input you need?
    b. How will you ensure that users enter valid input?
    c. How will you display the results to the user?
4. Decompose your program into the different components you need to incorporate into the final program. You will probably use Trello for this.
5. Throughout your development, you must trial multiple components. For example, this could include different ways to get user input, different ways of displaying output, etc.

    Important Note - for the 2.8 standard (Use advanced processes):

    You MUST trial multiple components and/or techniques and select the most suitable. This means trialling different ways to solve the same sub-problem and selecting the best. Even to get Achieved, you MUST have evidence of how you trialled AT LEAST TWO of your components and then gave reasons for choosing one over the other.

    NOTE that testing is about making sure something works. Testing and Trialling are NOT the same thing. Trialling is about trying out different ways of doing the same thing.
6. You also need to consider which advanced programming techniques will best make your program flexible and robust.
7. Select the best components and/or advanced programming techniques to include in your final program, based on the results of your testing and trialling.

8. Use your version control tools (e.g., GitHub) to save successive versions of your code and keep evidence of how you created the program in an ongoing manner (e.g., screenshots showing your file structure with appropriately named versions/program components, including brief annotations of the changes made in each version).
9. Ensure your testing includes both expected cases and relevant boundary cases (e.g., what happens when users get close to their budget). You may want to get other students or your family/whānau to test your program at each stage and provide feedback to help you improve your final program. Using others to test the program will help to ensure it is comprehensively tested for many different cases (including expected and relevant boundary cases). Note the improvements that could be made based on the testing and implement your changes.
10. Throughout the development of your program code, ensure that you document your program with appropriate variable/module names and comments that describe code function and behaviour. Follow the common conventions of your programming language (e.g. PEP8 and naming conventions or rules for program layout).
11. Comprehensively test your final program to ensure that it functions correctly and is of high-quality (e.g., bug free, has a well-presented and easy-to-understand instructions, contains all the required information).
12. Show how your program has addressed the relevant implications.
13. For Excellence you need to discuss how the information from planning, testing and trialling of the components of your program assisted you to develop a high-quality outcome.

What you will hand in:

Attach only these THREE things before submitting the assessment on Teams:

1. Your Evidence – using the provided PowerPoint template. This will include:
    a. A link to your **Trello** board – ensure this is a PUBLIC link
    b. A link to your **Github** project – ensure this is a PUBLIC link
2. A copy or link to the final version of your program.
   This is so that your teacher knows exactly which version to mark as your end product.
3. The marking guide – annotated to show the location of each item e.g., the slide number in your evidence document, or the file name of a particular component in your program.