Assessment Task Specification

Project Requirements

The human player and computer player take turns giving commands to their creatures. Both creatures begin with equal health points and energy points. The human player moves first, followed by the computer, and so on, until one creature or the other has no health points remaining. The victor is then declared and the game ends.

These requirements are fixed and cannot be changed.

Player and Computer Statistics

- Both player and computer start with 100 health and 100 energy.
- Both player and computer recharge 4 points of energy per turn.
- Both player and the computer have the same choices of actions during their turn:
    - Normal Attack
    - Special Attack
    - Re-Charge
    - Dodge
    - Heal

The effects of theses actions are detailed below.

Action Name	**Chance to Hit**	**Damage**	**Notes**
Attack	80%	1-10 Points	Standard strike on enemy. Requires 5 points of energy to use.
Special Attack	50%	5-20 Points	Uses the special attack. Requires 20 points of energy to use.
Re-Charge	N/A	N/A	The player does not attack this turn but instead recharges energy at 4 times the normal rate during this turn. As the creature is motionless during this time, the opponent's chance of hitting is increased by 10%.
Dodge	N/A	N/A	Attempts to predict the enemy attack and move aside.

			• Decreases enemy's chance of hitting by 30%, • Decreases energy charging rate by 50% during this turn.
Heal	N/A	N/A	Convert up to half of stored energy into health. Cannot exceed starting health. This action DOES NOT use up the player's turn allowing a second action, but the second action may not be Heal. Requires 10 points of energy to use.

A "round" consists of the human player choosing an action and the computer choosing an action, which their creatures in the arena then attempt to perform. To achieve a trivial pass, the program simply needs to play one round with appropriate display.

To pass, the game must:

- Loop until either the player or the computer has won
- Allow the user and computer each to choose from the different legal actions on their turn
- Implement simple logic for choosing the opponent's actions (i.e., not picking the same action repeatedly).

Project Report

The Project Report accompanies the programming project and contains the written documentation of your approaches to identifying and solving the assessment, and how your program implemented the solution. It must contain at least these 3 sections which correspond to the assessment marking criteria:

Problem Analysis and Design

To design this program you need to thoroughly analyse the requirements and decompose the overall problem into smaller problems which are individually solvable. You will need to consider exactly what the player will input to the program and what information the program will output to the player as well as what information it needs to store in order to function correctly.

Designs for your solutions must be demonstrated using appropriate tools, such as pseudo-code and flow diagrams, with stepwise refinement used to keep each at a reasonable level of detail. Compare your designs against the assignment brief and if you notice errors or omissions then make appropriate changes.

It is not unusual for the complete solution to develop incrementally as you alternate between planning and implementing. It is important that some program design is in place before implementation begins, and that you show how your design evolved over time to completely solve the problems. Keep track of changes to your design so they can be mentioned in the project report.

Things to consider:

- Storing/Updating health and attack information
- Displaying instructions and game status
- Capturing and validating input from the player
- Determining the computer opponent's move
- Determining win/lose conditions and declaring a winner
- Test that each action is functioning correctly

Implementation

Use the design to implement your solution in C++, with well-structured code that follows best practices. You should describe how your implementation proceeded from planning and design, and how any difficult obstacles were encountered were overcome.

The implementation should closely follow the design. If the design is found to be flawed once implementation begins, then a redesign should be thought out carefully and documented, describing what you had to change and why.

It is worth additionally checking that the final implementation fulfils the original assignment brief.

You DO NOT need to include source code in the body of the report, but you may want to include it in the appendix, especially if you want to refer to specific parts in the report.

Test Planning and Testing

Implementation should be tested as you develop the project, using the debugger, output values, test code, etc. Good testing does not mean just making sure the end product is free of obvious bugs – it means testing throughout development to ensure that as each component is completed, it and all previous code remains solid and functional. Use a range of tests that try to explore all the potential paths that the program can go down to confirm that it is valid.

Testing should also include a detailed final Test Plan that shows what functionality was tested, what the expected result was, and what result was actually obtained. Incorrect or unexpected results should be analysed, and tests and fixes should be documented. If the design must be updated because of a program error, the Design section in your report should reflect the changes.

Providing evidence that supports your testing is essential as it shows that you have followed through with testing, and that you understand how to correctly interpret and respond to test results. Evidence can include screenshots of the running program, screenshots of the debugger, error messages and references to any test code.

Extra Marks

Extra marks may be earned by adding functionality to the game and/or implementing the solution using more advanced techniques. Remember, the brief criteria MUST be met, so it is best to complete the basic project before attempting any extras. Some potential ideas might be:

Multiple battles against the computer with score kept.

Each player brings multiple creatures to the arena; the victor must defeat all opposing creatures to earn the victory.

Creatures may be of different types with different attributes.

The computer opponent may use an algorithm to determine which action to take, rather than random choices.

Using advanced programming approaches and constructs that lead to high quality code.

Implementing a menu system that allows the player to replay the game or choose between options.

Project Guidance

It is recommended that you do not focus on the creative/story potential of the project unless you have already built a functional game. Story, story-telling, and better game design are not assessed and therefore will not earn marks. You may choose to embellish the basic game with help text, dialogue, characters, or other customisations, but be careful not to compromise the intent of the assessment brief. Point values for damage, health and energy may be changed as well with the same limitations.

100 is a good number of health points and the basis for damage and other values used to describe the problem. You may choose different numbers provided the results are similar.

50 is a good number of energy points to start with, assuming that the special attack consumes all 50 points.

Check the unit web page for additional information and responses to questions which emerge during the semester.

Hand-in Details

The Game Project and Project report must be submitted online before the deadline to receive full marks. Your submission must contain two files:

1. Your Project Report document, containing the design, testing, conclusions and any supporting diagrams, tables or charts.

   - The front page of this report should be a title page that contains at least the information **"[Student Number]_[LastName]_[FirstName]_DAC416_AE1_Report".** *(square brackets indicate placeholders, they should not be in the final file name)*

   - The document should preferably be in either MS Word or .pdf format.

   - Your source code may optionally be included as an appendix in the report.

2. A compressed/zipped file containing your Game Project

3. Only use .zip files, not .rar or other formats. If I can’t open your project I can’t mark it.

4. These two files should be submitted individually, not in one zip file, *only* the project file should be zipped.

5. Name the file "[Student Number]_[LastName]_[FirstName]_DAC416_AE1_Project", *(square brackets indicate placeholders, they should not be in the final file name)*.

6. Include all of solution and source code files: .h and .cpp files and any additional files you have created as resources for you project.

7. Make sure you include all source and resource files necessary to build and run your program, including any files supplied to you.

8. You do not need to include temporary or intermediate files created during the build process.

9. Due to the volume of network traffic especially near deadlines, the online submission system may be slower than you expect. Extensions for connection problems will not be granted unless university-wide. Give yourself plenty of time!

10. You MUST keep an exact copy of your project as backup in case of submission failure.

11. Ensure that the PC used to submit your work or create your submission package is free of viruses or malware. Submitting digital media containing any form of malware will result in significant penalty or failure.