MCD4720 - Fundamentals of C++

Assignment 3 - Trimester 3, 2018

Submission guidelines

This is an individual assignment, group work is not permitted

Deadline: January 22, 2018, 11:55pm

Weighting: 25% of your final mark for the unit

Submission Instructions:

A zip file containing your Visual Studio project and the associated documentation files (project plan) must be compiled and uploaded to the Moodle site. Your code MUST be submitted as a Visual Studio project to facilitate ease of assessment and feedback.

Late submission:

- By submitting a Special Consideration Form or visit this link: https://goo.gl/xtk6n2
- Or, without special consideration, you lose 5% of your mark per day that you submit late (including weekends). Submissions will not be accepted more than 5 days late.

  This means that if you got *Y* marks, only $(0.95^n)\times Y$ will be counted where *n* is the number of days you submit late.

Marks: This assignment will be marked out of 50 points, and count for 10% of your total unit marks.

Plagiarism: It is an academic requirement that the work you submit be original. If there is any evidence of copying (including from online sources without proper attribution), collaboration, pasting from websites or textbooks, **Zero marks** may be awarded for the whole assignment, the unit or you may be suspended or excluded from your course. Monash Colleges policies on plagiarism, collusion, and cheating are available here or see this link: https://goo.gl/bs1ndF

Further Note: When you are asked to use Internet resources to answer a question, this **does not mean copy-pasting text** from websites. Write answers in your own words such that your understanding of the answer is evident. Acknowledge any sources by citing them.

Task Details:

This assignment consists of one main programming task. The purpose of this assignment is to have you design and implement an object-oriented program in C++, as well as reflect on your approach and design choices. The assignment comprises the following components:

- A diagram with annotation that describes your object-oriented design
- The completed program
- A 300 word reflection on your program

Successful completion of the fundamentals of the task as described may obtain you up to a maximum of 80% of the total assignment marks. The last 20% of the mark will be allocated to additional functionality that you can design. The additional functionality should demonstrate advanced or more complex application of principles covered to date. It need not be large amounts of work but should demonstrate a willingness to explore new and advanced concepts. You must detail what you have done in an accompanying "readme" file.

The assignment must be created and submitted as a Visual Studio 2017 project. You may complete the exercises in your preferred IDE, however you should create a Visual Studio project in order to submit. This project must then be zipped up into one zip file for submission named "***YourFirstNameLastNameA2***.zip". This zip file must be submitted via the Moodle assignment submission page.

- Explicit assessment criteria are provided, however please note you will be assessed on the following broad criteria:

- Meeting functional requirements as described in the assignment description
- Demonstrating a solid understanding of C++ concepts, including good practice

- Demonstrating an understanding of specific C++ concepts relating to the assignment tasks, including object-oriented design and implementation and the use of Pointers\
- Following the unit Programming Style Guide
- Creating solutions that are as efficient and extensible as possible
- Reflecting on the appropriateness of your implemented design

NOTE! Your submitted program MUST compile and run. Any submission that does not compile will be awarded zero marks. This means you should continually compile and test your code as you do it, ensuring it compiles at every step of the way.

If you have any questions or concerns please contact your lecturer as soon as possible.

Assignment Task 3: Hero's Quest

In this assignment, you are to write an action adventure game, where your hero character will fight a collection of computer-controlled monsters in the grand colosseum. Defeat them all, including the boss, and you will become the grand champion!

Basic Game Play

In this program, you will control a hero character. Your hero character will fight a series of monsters. By beating a monster, you will earn extra stats and prize money. Eventually you will fight the grand boss monster. Beat the all the monsters and boss without dying and you will become the grand champion! The play will go as follows:

- You are the hero. You have a name (user defined), health (can be between 0 and 50, starts at 20), attack (starts at 3, max is 10), defence (starts at 3, max is 10), and special attack (starts at 2, max is 10). The hero also has an amount of prize money (starts at 0 and has no upper limit).
- In the game there are 5 monsters waiting to fight. They will fight in order. There are 4 regular monsters and finally one boss monster.
- The regular monsters have a name, health (can be between 0 and 100), attack (can be between 0 and 10), and defence (can be between 0 and 10). They do not have a special attack attribute.
- The boss monster has the same attributes as the regular monster but also has a special attack (can be between 0 and 10)
- You will then fight the monsters in order. The fighting process is described separately below.
- When you win a fight, you earn 20 health points, and a random amount of attack, defence, and special points. For each of these you could earn 0, 1 or 2 points. For example after one successful monster battle you might earn 1 attack, 1 defence and 2 special. You also earn prize money, which is how much health you have left at the end of a battle (before adding the 20 health points for winning). This means you earn more prize money if you get less damage during the fight.
- If at any stage your health goes to 0 the game is over and the monsters win.
- If you defeat all monsters and the boss then you win and the game is over.

Monster Stats:

Below are the stats for each monster (health, attack, defence and special for the boss):

- Monster 1: 10 health, 1 attack, 1 defence
- Monster 2: 20 health, 3 attack, 2 defence
- Monster 3: 30 health, 5 attack, 4 defence
- Monster 4: 40 health, 6 attack, 7 defence
- Boss: 50 health, 8 attack, 8 defence, 5 special

Battle:

When you battle a monster, the logic is as follows:

1. The hero generates a random number between 1 and 6. It is added to the skill level.
2. Another random number 1-6 is generated for the monster and added to its defence level.

3. If the player's attack total is higher than the monsters defence total, the monster loses the difference between the two values in health. If the monsters total is equal or higher, no health is lost.
4. Steps 1 to 3 then occur except with the Monster attacking and Hero defending.
5. Steps 1 to 4 keep continuing until either the player or monster runs out of health.

For example, here is a case with the Hero attacking and Monster defending:

- Hero generates a random number 1 to 6 and adds it to attack. For example, they get a 4 and add it to their attack, which is 3, giving them a total of 7.
- Monster generates a random number 1 to 6 and adds it to defence. For example, they get a 3 and add it to their defence, which is 1, giving them a total of 4.
- As the hero's score is higher they win the round. They then take 7 – 4 = 3 points off the health of the monster.

Special Attack:

- The hero may launch a special attack once per monster fight.
- If the player chooses to do a special attack, they can add their special attack value to their overall attack score, giving them an advantage.
- The player must choose before the attack section of each fight if they wish to do a special attack. They can then not choose to do it again for the rest of a monster fight.

Extra Functionality

The marking criteria indicates that you should make some individual additions to this in order to achieve the final 20% of the mark. This is up to you. You should aim to add some additional creative elements to the gameplay, as well as advanced object-oriented design elements or advanced use of pointers.

Program Design Diagram

In this final assignment you are able to design your program in any way you like. You must provide a diagram with annotation that shows the classes you have in your program and how they interact.

- It should contain any relevant notes that describe the classes.
- This diagram should use the UML notation that has been demonstrated in the labs.

Please use the following links to learn UML:

- http://icarus.cs.weber.edu/~dab/cs1410/textbook/9.Classes_And_Objects/uml.html
- https://cppcodetips.wordpress.com/2013/12/23/uml-class-diagram-explained-with-c-samples/

Program Reflection

You must provide a 300-word written reflection of your object-oriented design and how well you believe it was to implement. You should cover the following areas:

- Why you designed it the way you did.
- How well you were able to code it, highlighting any issues you found once you tried to implement your design.
- How you might change your design to make your solution easier to implement, more efficient, or better for code reuse.

Suggested Development Process

Below is a suggested process you might use to develop your assignment piece by piece so that it is not too overwhelming. Feel free to follow this if you think it will help (or not!):

1. Design the program on paper… decide what classes you think you need and how they will relate to each other. Write down your thoughts on why you think this is the best design. You need to do this for the assignment anyway! Don't forget how you can use inheritance and pointers.
2. Write a simple class that you need and test it!
3. Write another simple class that you need and test it!
4. Do these two classes interact? If so, see if you can get basic interaction happening!
5. Repeat the process for each additional class or function… either add to existing classes or write new ones if you need to. Remember, just focus on the small specific task!

I hope this helps. Important thing is to focus on small sub-tasks one at a time and build up functionality that way.

Assignment 3: Marking Criteria [up to 100 marks in total]

- Does the program compile and run? Yes or No
    - **Zero marks will be awarded for a non-compiling program.**

Class Design Diagram [10]

- Does the diagram clearly show all the different classes used in the program and each class' attributes and methods? [5]
- Is rationale provided for the classes chosen and overall design? [5]

Functionality [35]

- Game set up, including setting up the player and creating a collection of monsters [5]
- Appropriate game dialog and player interaction [10]
- Appropriate battle mechanics [5]

- Implementation of Special Attack [5]
- Correct end game conditions applied [5]
- Appropriate display to the screen, user interface, and basic data validation [5]

Quality of Solution and Code [25]

- Has a well-designed OO program been implemented (i.e contains classes appropriate to the assignment brief)? [5]
- Does each class, as written, implement just the things required by the class (i.e the classes do not include things not specific to the class)? [5]
- Is there appropriate use of pointers in the program? [5]
- Does the program perform the functionality in an efficient and extensible manner? [5]
- Has the Programming Style Guide been followed, including documentation, code laid out properly in cpp and header files. etc.? [5]

Extra Functionality [20]

- Does the program addition demonstrate advanced application of programming concepts? [10]
- Does the program addition demonstrate functional creativity? [10]

Reflection [10]

- Discussion of motivations of the program design [3]
- Discussion of how well the design was to implement [3]
- Discussion of what they would do differently if they were to start it again [4]