CSCI 1300 CS1: Starting Computing

Instructor: Ashraf/Corell/Cox/Fleming, Fall 2019

Project 3: Choose Project, Meet with TA/CA to go over design
Before Wednesday November 13

Project 3: Submit Class files & Code Skeleton
Due Wednesday November 13 by 11 PM (no early submission bonus)

Project 3: Final Deliverables - Due Wednesday December 4 by 11 PM (no early submission bonus)

Project 3: Project Report - Due Sunday December 8 by 11 PM (no early submission bonus)

Project 3: Interview Grading
Begins on Thursday, December 5. Must be completed at the latest on Friday December 13.

This project is worth 15% of your overall course grade.

Create Your Own Project

For the Final Project you will implement a **turn-based, text (adventure) game** in C++. The minimum requirements for the project can be found below. From known games, suggested examples include *Where in the World is Carmen Sandiego?, Final Fantasy,* and *MUD.* Of course, you are encouraged to come up with a novel and unique idea in order to distinguish your project from the others. Remember, your point of comparison should be the "**The Pokemon**" Project (see the write-up for that on Moodle).

Not recommended: Battleship, Tic-tac-toe, Sudoku. This list is by no means exhaustive. If you can google your desired game and a full C++ implementation shows up, you should choose something else.

Minimum Requirements:

Your implementation of the project should have:

- At least 4 user-defined classes
- At least 2 of these classes should have 4 or more data members.
- At least 1 class must include an array of objects from a class that you created.
- Appropriate methods for each class (including getters and setters)
- Your project implementation must include at least:
    - At least 6 **if** / **if-else** statements
    - At least 4 **while** loops
    - At least 4 **for** loops

    - At least 2 nested loops (these count in the 4 **while** and 4 **for** loops above)
    - At least 7 strings variables/data members
    - Reading from files
    - Writing to a file
- Your project must have an interactive component (ask the player for input, create menus for choices, and so on). In other words, it's a *game*!
- You must display stats at each turn. It helps you debug, and it helps the user interact with your game in a more meaningful way.
- As part of the interactive component, you project must include a **2D map**; the characters involved in the project interact with one another and make decisions based on their environment. You must display the map at each turn.
- Your project implementation must include:
    - At least 5 menu options (aside from Quit/Exit)
    - At least 2 of these options must have a second layer of menu options
    - At least 2 menu options (primary or secondary layer of the menu) should include a random component, at least one each from the following:
        - The value of a variable is selected at random from a certain range of values (I.e. select a value at random between 1 and 6).
        - A probability value determines one of the outcomes (I.e. there is a 60% chance a certain event will occur).

Very important:

Think of the game you are developing as a real application. Try to anticipate user errors and respond accordingly. For example, if the user can only choose between 4 menu options but they choose option 5, your game should not crash, but instead display appropriate error messages and ask the user to choose again. Other examples of user error: trying to explore locations outside the map size or entering a negative parameter size. In general, when testing your program, it is recommended that you try to "break" it. Fix all the instances that would cause errors or unwanted game outcomes, so that during the grading interview your program is unbreakable.

Extra credit (10 points total possible):

#1: (3 points) Implement a sorting algorithm (write your own implementation - do not use a Library function or any outside resources) and apply it to a task in your program.

#2: (4 - 7 points) Present your project in a video, or in class (see below)

Project timeline:

Checkpoint 1: Choose Project, meet with TA or CA before Wednesday, November 13

You will not only need to come up with an idea for your project, you will also need to come up with your own program structure. You will need to decide how the information will be stored in objects and how it will be passed between objects. You need to choose which classes to define, what are their data members, and which class/object is responsible for each part (functionality) of the game.

The choice of classes is entirely up to you. But to ensure you did not choose an approach that is not feasible, we require you meet for 15 minutes with the professor, a TA, or a CA, to go over your classes. **This meeting is mandatory**. You must choose a slot from the **Moodle scheduler** and have the meeting before Wednesday, November 13 at 6 PM. This meeting will count as interview grading in terms of the no-show and (re)scheduling policies. See the **syllabus** for more information.

Failure to schedule and attend the Project 3 Design Meeting will result in a 10-point penalty on your Project score.

Note: Even if you choose to do "Pokemon" project, the design meeting is mandatory. Follow the instructions for the Project Proposal in the *Pokemon* Project write-up.

Checkpoint 2: Submit class files & Code Skeleton via Moodle, due Wednesday November 13 @ 11 PM

Your .h files should be *complete* with all the data members and member functions you will be using for each class. For the class implementation .cpp files, you should fully implement simple functions like your getters and setters. For more complex functions you can include function stubs with detailed comments. At a minimum, the input parameters, output type and pseudocode description should be present.

For example, if we were stubbing a function to implement bubble sort and return the number of swaps we might give in our code skeleton:

```
/*
   1. Compare adjacent elements. If the first is greater than the
      second, swap them.
   2. Do this for each pair of adjacent elements, starting with the
      first two and ending with the last two. At this point the last
      element should be the greatest.
   3. Repeat the steps for all elements except the last one.
   4. Repeat again from the beginning for one less element each time,
      until there are no more pairs to compare.
*/
```

```
int bubble_sort(int arr[], int size)
{
      int swaps = 5;
      return swaps;  // function returns expected type (int)

}
```

Your Code Skeleton should be inside your driver file and it should contain detailed comments with pseudocode explaining the functionality of the project.

You must submit your Class Files and Code Skeleton to Moodle to get full credit for the assignment. Failure to submit the Class Files and Code Skeleton will result in a 10-point penalty on your Project score.

Checkpoint 3: Final Deliverables due Wednesday December 4 @ 11 PM

The final version of your project will be due on **Wednesday December 4 @ 11 PM** (no early submission bonus available and there will be no extensions). You must submit a .zip file to Moodle which includes:

- All .h and .cpp files, including the main driver program, correctly indented and commented.

Checkpoint 3A: Project Report - Reflection Activity, due Sunday December 8 @ 11 PM

Write a 1-2 page report containing answers to the following **Reflection questions:**

1. How did you prepare for the Project?
2. How did you develop your Code Skeleton? In what way(s) did you use your Code Skeleton?
3. Reflect on how you could have done better, or how you could have completed the project faster or more efficiently.
4. In addition, write a paragraph answering the following question, in the context of the Project in CSCI 1300:

   Did you have any false starts, or begin down a path only to have to turn back when figuring out the strategy/algorithm for your Final Project program? Describe in detail what happened, for example, what specific decision led you to the false starts, or, if not, why do you think your work had progressed so smoothly. In either case, give a specific example.

Note: all reflection papers must be individual.

Submit a PDF file of your report to the Moodle dropbox by **Sunday December 8 @ 11 PM**

Failure to submit the Report will result in a 10-point penalty on your Project score.

Checkpoint 4: Interview Grading - will begin on **Thursday December 5**, and must be completed at the latest on **Friday December 13** (reading day).

Checkpoint 5: Extra Credit Opportunity #2- up to 7 points (due Sunday, December 8th @ 11 PM)

- Make a 5 minute (+ or - 1 min) video explaining:
  - The project idea
  - Implementation and approach
  - A demonstration of the working project
  - A **Google form** will be made available to **submit a link to your video**.
- **OR** you can present your project in lecture (5-7 min presentation).
  - You must **sign up to present by Wednesday, December 4 @ 11:55 PM**
  - Limited slots will be open and will be a first come, first serve
  - Presentation should include:
    - project idea,
    - implementation and approach, and
    - a demonstration of the working project.
- You **cannot** earn extra credit for doing both the video and the presentation in class
- All students who present will receive 4 points. The best Create Your Own projects who present their game will receive an additional 3 points.

Collaboration:

All work for this assignment (and course in general) must be your own, original work. You may work together, but your assignment submission is **your own**. Your work may not include code taken from online resources like Chegg or StackExchange, or from other students (past or present), even with modification. Any such instances constitute Academic Dishonesty (passing off others' work as your own) and will earn you a 0 on the assignment and a trip to the Honor Code Council. If you aren't sure if something is okay, then please just ask!

Points:

Note: <u>If your code does not compile, you cannot score above 40 points for the project.</u>

20 points - meets minimum requirements specified on the first page (# of classes, loops, etc.)

40 points - interview grading

- TA's questions about your project
- algorithms descriptions, comments, good style
- code compiles
- if your code does NOT compile, you can get at most 20 points from meeting the minimum requirements, and at most 20 points from the interview

40 points - project functionality

- the game plays at outlined in the project description
- your solution accounts for user error

3-10 points extra credit

- (3 pts each) Your game has a sorting feature
- (4- 7 pts) Create a 5 min (±1 min) video explaining your project implementation & demo, ***or***
- sign-up to present and demonstrate your project in lecture (5-7 min presentation).

Possible Deductions:

- 10 points for not attending the project meeting (Checkpoint 1)
- 10 points for not submitting code skeleton and class files (Checkpoint 2)
- 10 points for not submitting report (Checkpoint 3A)