COP 3331: Object Oriented Design
Project 1
Due June 11th at 11:59 P.M.

You are not allowed to use the Internet. You may only consult approved references[*].
This is an individual project.
This policy is strictly enforced.

Project Objectives

To learn more about

- Object modeling based on project requirements
- Class creation
- Using objects
- Basic user interfacing

Submission Guidelines

- A zipped Visual Studio solution submitted through Canvas. We will be using VS 2013 for this course.
- Points will be deducted for not submitting a solution.
- **Late submissions will be accepted for 24hrs after the due date, but with a 20pt reduction on the final grade.**

For full credit, the code that is submitted must:

- Have a comment block including your name.
- Be implemented in a file using the specified file name, if applicable.
- Be readable and easy to understand. You should include comments to explain when needed, but you should not include excessive comments that makes the code difficult to read.
  - Every class definition should have an accompanying comment that describes what is for and how it should be used.
  - Every function should have declarative comments which describe the purpose, preconditions, and postconditions for the function.
  - In your implementation, you should have comments in tricky, non-obvious, interesting, or important parts of your code.
  - Pay attention to punctuation, spelling, and grammar.
- Have no memory leaks.
- Use no magic numbers. Use constants in place of hard-coded numbers. Names of constants must be descriptive.

[*] You may use all class material and the C++ reference sites posted at the beginning of the semester. Any additional resources must be pre-approved. You must cite all references used.

Some additional coding guidelines which will help you to create efficient, organized code (optional)

- Using #define header guards to prevent multiple file inclusion. The typical form of the symbol name is `<FILENAME>_H_`
- Limit the amount of code you copy and paste. If you need to reuse a section of code, you should create a function to place it in instead.
- Define functions inline only when they are simple and small, say, 5 lines or less
- Function names, variable names, and filenames must be descriptive. Avoid confusing abbreviations.
- It is best to use spaces instead of tabs. Most IDEs and text editors will give you the option to insert spaces in place of tabs (typically 4).

Project Description

You will be creating a terminal-based box office kiosk program to be used by a major entertainment company, Starlight Entertainment. Starlight owns several different venues and wishes to install these automated kiosks at each location. Each kiosk is capable of selling tickets to all Starlight venues as well as food and drinks. You will be designing the program which patrons will interface with to make ticket, food, and drink purchases. For now, the bigwigs want a basic system to get them up and running quickly. The requirements are listed below

Project Details

For this project, you are required to create *at least* the following two classes. Each of these classes represent items that are sold at the kiosks:

- Event taking place at Starlight venues. An event must have the following information
  - Name
  - Description
  - Price
  - Event time
  - Event duration (Minutes)
  - Number seats available
- Concession item (food or drink). These delicious items must have the following information
  - Name
  - Description
  - Price
  - Size (Small, Medium, Large)
  - Number available

You may choose to create any additional classes that you feel are needed to properly model the problem at hand. The two classes listed above and the information needed for each are the *minimum* required for this assignment; if you do not include one or both of these, you will lose points. Your program will utilize these classes to build an inventory of event and concession items to be put on sale at the kiosk on a daily basis.

In addition to the creation of the kiosk inventory, a basic user interface must be built which allows a patron to

- Browse available event and concession items
- View information about an event/concession item (listed above)
- Add/Remove items from cart
- Add/Remove multiple of an item (based on availability)
- Purchase items in cart
- End their session without buying anything (emptying cart)

In order to properly support all user actions listed above, your program must also be able to

- Have at least 5 different concession items available for sale
- Have at least 2 different events for which tickets may be purchased
- Track concession stock and available # of seats for each event (out of stock items may not be purchased)
- Have ability to set different prices on items:
    - Events may have different prices
    - Drinks, popcorn, and candy **must** come in 3 sizes (small/medium/large) with prices set according to size
- Set/Get values for each piece of info for event/concession items
- Get input from user (Y/N, # of items to buy, etc)

Project Assumptions

For this project, you may make the following assumptions

- Events are only for a single day; tickets cannot be purchased for future events
- Ticket prices are equal for all seats in a given event
- All other concessions besides those listed above may be sized and priced however desired
- Payment is charged automagically (simply display total charge on checkout and ask user to accept)
- There are no returns