**Programming II – Program 8 (Final Project)**

**Due Fri, April 20**

**Spring, 2018**

**(50 pts)**

* **No Late Work Accepted**
* **Individual Project (not a group project)**

You are to implement a simplified version of the classic "space invaders" game. I am less concerned about the exact functionality and appearance of the game than I am that you exercise appropriate design and implementation skills.

Specifically, your solution must include the following:

* Appropriate use of multiple classes using composition and/or inheritance where appropriate. Do not force inheritance; use it only if it clearly applies and makes things easier.
* Appropriate use of member functions and data.
* Appropriate distribution of data/functionality to classes. Each class should be defined in a separate source file. If you main() function is huge, this is a good indicator that things haven’t been distributed in a reasonable way.
* You must use the STL **list** class where appropriate (this implements a linked list). On bboard, I’ve zipped up a project (vectorEraseDemo) that contains two projects:
  + vectorEraseDemo – this illustrates a common bug associated with erasing things from vectors, and how to fix that. I want you to understand this issue, but we won’t be using vectors for this particular program because they aren’t the best fit for the problem.
  + listEraseDemo – this illustrates how to use the STL list class, including how to delete (erase) things from the list
* Appropriate design and comments in the code

**Functional Requirements**

* Your game must begin with at least 10 aliens (a single row of aliens is sufficient).
* The game must have a 'Start' “button” that begins the action.
* The player moves the ship (horizontally only) using left/right arrow keys. The space key is the 'fire' button.
* You should be able to fire your ship’s gun such that you have multiple missiles en route simultaneously.
* Periodically, a *randomly* selected alien should drop a bomb. The interval at which bombs drop should vary between a minimum of one second and a maximum interval (you can decide on the maximum interval). Multiple bombs may be en route simultaneously.
* The row of aliens should move down the screen toward the ship (you can decide the speed). A direct line of descent is satisfactory (i.e., aliens don't have to wiggle down the screen). If any alien reaches the level of the ship, count this as a destroyed ship (a lost life), and restart the level. You should establish a limited number of lives per level.
* The program must detect hits on aliens and the ship. An alien simply disappears when it is hit (of course, the missile disappears also). When the human ship is hit, the number of lives counter should be decremented, and the state of the game is reset to the beginning of the current level (you can decide how many lives the human gets). If the number of lives is exhausted, the game is over. A message should appear on the screen showing the winner. At this point, the Start button should be available; the user will click Start to play again.
* If the player destroys all of the aliens on the initial level, you should re-populate with a different kind of alien. Level 2 aliens should appear differently than level 1 aliens, and should drop bombs differently than level 1 aliens (e.g., you could make the bombs bigger or faster or more frequent than Level 1 aliens). You are only required to have two levels. If the user kills all Level 2 aliens, the user wins the game.
* Labels on the screen should be used to display the following information:
  + Number of lives left for the gunner.
  + Number of destroyed aliens.

**Extra Credit**

* Provide a way for the user to pause and resume the game. While paused, the user should be able to save the state of the game to a file or exit the game. The next time the user runs the program, if the file exists, it should ask the user whether they wish to load the game or start a new game (this can be done in the console). The name of the file can be hard-coded.
* Shields for the hapless human. Shields should be gradually destroyed when hit by a bomb.

**Header File Warning:**

* Be careful to avoid a mutual-dependency between header files (e.g., where A.h includes B.h, and B.h includes A.h). If you are tempted to create such a dependency, the solution is typically to use a *forward declaration* of a class in one of the header files (rather than #include the other header file). See the discussion on page 821, as well as examples on page 820-821.

**Turn-In Instructions**

* *You must add the following information to the Readme.txt file in the project:* 
  + *Any special instructions for running the game*
  + *List of incomplete functionality or known problems/bugs. I will penalize less for known problems than for problems I discover during testing that you did not document.*
  + *A description of any ‘extras’ you added to the game.*