**CSCE 2014 – Programming Project 2**

**Midpoint Due Date – Feb 15, 2017 at 11:59pm**

**Final Due Date – Feb 22, 2017 at 11:59pm**

**1. Problem Statement:**

The goal of this programming project is to give students experience with linked lists while implementing part of a creature fighting video game. If you have ever played a creature fighting video game like StarCraft, you know that the number of creatures in the game varies significantly over time, and they move all over the board in semi-random directions fighting each other. Hence, a linked list is ideal for storing information about all of the creatures that are currently alive. We can add creatures to the linked list as they are spawned, and we can delete creatures from the linked list when they are killed. The game engine can also use the linked list to control how the creatures move, what creature battles take place, and what creatures to display on the screen.

For this programming project, students must design, implement and test a “CreatureNode” class that contains typical information about a single creature in a video game. Then, students must design, implement and test a “CreatureList” class that contains a linked list of CreatureNode objects. Both classes will have a collection of specific methods that will be useful for video game developers when creating a creature fighting video game.

Your “CreatureNode” class should store the following information:

* ID – a unique identifier for the creature
* Xposition, Yposition – the creature’s current position
* Xvelocity, Yvelocity – the creature’s current velocity
* Health – some measure of how healthy the creature is
* Strength – some measure of how strong the creature is

Your “CreatureNode” class should implement the following methods:

* Constructor – initialize one creature
* Destructor – uninitialized one creature
* Getters – set creature information
* Setters – get creature information

Your "CreatureList" class should store the following information:

* List – a linked list of CreatureNode objects
* Count – the current number of creatures in the linked list
* Xmin, Xmax – the range of X positions for creatures
* Ymin, Ymax – the range of Y positions for creatures

Your “CreatureList” class should implement the following methods:

* Constructor – create an empty linked list
* Destructor – delete all nodes from the linked list
* SpawnCreature – insert creature node into linked list
* KillCreature – delete creature node from linked list
* BlastCreatures – damage all creatures near blast position
* MoveCreatures – move all creatures according to their velocities
* PrintCreatures – print information about all living creatures

It is beyond the scope of this project to design, implement and test a real video game using your CreatureNode and CreatureList class. Instead, you will be given “skeleton” implementations of the two classes above and a simple main program that makes a sequence of calls to the methods listed above to verify that they are working correctly.

**2. Design:**

Your primary design tasks are: (1) to select the data types for all of the private variables in the creature classes, (2) to specify the parameters for all of the creature methods, and (3) to work out the algorithms you will use to implement all of the methods. For the basic linked list operations (create, insert, delete, print), you are encouraged to look at the textbook, the class notes, and sample source code for inspiration.

For the video game specific operations (blast, move) think about how creatures would be effected in a video game if there was an explosion at location (bx,by) and the creature is at location (x,y). How much damage would occur? What should you do if the creature’s health falls below zero? Similarly, for creature motion, where would you expect the creatures to be located if they are at location (x,y) and they have a velocity of (dx,dy)? The tricky part here is deciding what to do if the creature reaches the screen boundary. Do you want to have them “bounce off” the screen boundary and go in the opposite direction, or do you want them to “fall into the void” and die? Explain all of these design decisions in your final project report.

**3. Implementation:**

To simplify the implementation process, you will be given skeleton class definitions in “CreatureNode.h” and “CreatureList.h” and skeleton implementations in “CreatureNode.cpp” and “CreatureList.cpp”. The file “main.cpp” contains a collection of calls to these classes that demonstrate typical operations in a creature fighting video game. If you put all of this source code in one directory, you can compile your program using:

g++ -Wall \*.cpp -o hw2

Since you are starting with skeleton class definitions, your main task is to implement each of the methods. It is always a good idea to work incrementally, writing comments, adding code, compiling, debugging, a little bit at a time. Once you have the methods implemented, you can add calls to the methods to your main program to complete your project.

Remember to use good programming style when creating your program. Choose good names for variables and constants, use proper indenting for loops and conditionals, and type clear comments while you are writing the code. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

**4. Testing:**

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

**5. Documentation:**

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

**6. Midpoint Project Submission:**

To encourage students to get an early start on their programming project, students are required to upload into Blackboard a partial solution to their programming project on the midpoint due date shown above. The program does not need to be complete, but it must compile and perform some of the tasks listed above. This midpoint solution is worth 10% of your final grade on the project.

**7. Project Submission:**

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

* 10% off if less than 1 day late,
* 20% off if less than 2 days late,
* 30% off if less than 3 days late,
* no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

**8. Academic Honesty Statement:**

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.