CSC 260 Fall 2012

Assignment 6 – Dynamic Memory Management

Due: Tuesday November 27, 2012 by midnight

Grade As Per Rubric Provided: 20 points, late penalty 10% per day

Objective:
The objective of this assignment is to internalize through practice and experience, important concepts related to dynamic memory management in C++.

Requirements:
Assume that you are a software engineer at GF Software Solutions, Inc. and have been given a choice of projects to work on. Regardless of which project you choose, your program must have a well-designed class that extensively demonstrates dynamic memory management with either a dynamic array or a linked list. You may not use the C++ standard template library.

Assume that any classes you design will be used by you or your colleagues for other projects, i.e. they should be **reusable**. You must design and implement "safe" classes, i.e. memory management must be handled appropriately to prevent memory leaks. As always, you should follow the principles of information hiding, encapsulation and responsibility-driven design, i.e. all functionality and error handling must be provided by the appropriate classes. It is not sufficient for the program to just "work". Rather, it must meet all the requirements, handle errors gracefully, and use appropriate constructs and algorithms. There should be some thought given to why a particular approach would be more efficient or practical than another approach. Document this in the design and/or code where relevant.

The driver, i.e. the `main()` function should instantiate objects and invoke the appropriate methods needed to demonstrate that your classes meet the above requirements.

Review the projects below and select **one** to analyze, implement and test.

Project A: Be Creative.

Propose a project that interests you, that will use a dynamic array or linked list and meet all the requirements specified above.

Submit for approval in SOCS Dropbox "Assignment 6", a coherent, typed proposal with sufficient details of the specifications by midnight **November 13, 2012**. Projects without prior approval will not be accepted.

Project B: Polynomials.

Modify the Polynomial class you implemented for Assignment 4 (non-template version) so that it uses dynamic arrays instead of the fixed size arrays.

Project C: Sets.

You have been assigned to design and develop the **`Set`** class as defined in math.

A 'set' is a collection of unique entities that share some common properties. Elements in a set are not ordered. For this assignment you can assume that the `Set` has positive, even integers, including 0.

The `Set` class uses a dynamic array or a linked list to store the elements in the set.

The services that must be provided include:

- Add an element to the `Set` – an element can be added only if it is a positive even integer and not already in the `Set`.
- Remove an element from the `Set` – an element can be removed only if it exists and if the `Set` is not empty.
- Check if the `Set` is empty; returns true or false.

- Overloaded << operator that displays the elements of the `Set` in proper notation. For example, a `Set` that contains the elements 4, 2, 12, 8, and 16 is displayed as
  `{4, 2, 12, 8, 16}`
- Overloaded >> operator that allows the user to input elements for the `Set`; input may be from the keyboard or from a file.
- Overloaded + operator that returns a Set that is the 'union' of two `Sets`; all unique elements in the two `Sets` are included in the resulting `Set`; duplicate elements are discarded.

General Instructions:
Before starting to work on this project, review Chapter 14 in Dale & Weems, Slide Sets 13 and 15 on the wiki at http://tcnjcsc340.pbworks.com/w/file/32331216/SS13%20DynamicMemory.ppt.pdf and http://tcnjcsc340.pbworks.com/w/file/33323045/SS15%20Linked%20Lists.ppt.pdf, and the class exercise on pointers.

Implement the program in C++ following best practices for software development. See the "Guidelines for programming assignments" at http://tcnjcsc340.pbworks.com/w/page/34555300/Guidelines-for-programming-assignments.

Compiling and Testing:
Create a '`make`' file to compile your code. For details and instructions refer to the wiki page at http://tcnjcsc340.pbworks.com/w/page/59486008/Make%20And%20Other%20Build%20Tools.

The executable should be named `assign6`***`INIT`***. Replace ***`INIT`*** with your initials.

The assignment must be completed on time and the program compiled on the iMac operating system (machines in Holman 370). Use the naming conventions specified in this document. Additionally, submit a `readme`***`INIT`***`.txt` file with instructions for correct execution and the input files you used for testing your implementation.

Deliverables:
Zip together and upload to SOCS in the Dropbox folder 'Assignment 6' by **November 27**:

- Source code for the program.
- Script showing successful testing.
- Any text files used to input data.
- Approved proposal with any modifications, if you chose Project A.