CS 474: Object Oriented Programming Languages and Environments

Fall 2014

C++ project

Due time: 7:00 pm on Saturday 12/6/2014

This project requires you to write *set calculator app* in C++. This application is similar to the first Smalltalk project and to the Objective C project that you just completed. This time around, you will define an abstract superclass Set with two concrete subclasses for managing sets using different data structures. In particular, subclass SetAsList will implement sets as linked lists whereas subclass SetAsOC will use an ordered collection class. You must code both the linked list and ordered collection classes yourself; do not use predefined classes or library classes for these two implementation of sets.

As with the Smalltalk project, your application must work with two sets, which we will call *A* and *B* here. For simplicity the sets contain integer numbers. No duplicate values will be allowed in each set, regardless of the chosen set implementation.

Implementation notes. Your program will perform line input and output without a graphical user interface. Sets *A* and *B* must be bound to polymorphic identifiers of type Set*. Operations *union* and *intersection* must be implemented in superclass Set. You must use a *virtual destruction* and *virtual copying* scheme when deleting and copying sets. In addition to the commands below the programmable interface of class Set should support an indexing operator *operator[]()* and a size method *size()*. The indexing operator takes as input an integer *i* and returns the element at the *i*-th position in the receiver (a set instance). The size operator returns an integer indicating the number of elements currently stored in the receiver. Both methods are declared in the abstract superclass and implemented in the two concrete subclasses.

Use a command line interface for entering the commands below. Your command line interface will prompt the user for a command, and then execute the command. Here is a list of commands. Make sure not to cause any memory leaks or dangling pointers in the implementation of these commands.

1. l (lowercase 'el')— Initialize sets as lists. This command allows interactive users to reset the calculator and initialize two new sets *A* and *B* using the linked list implementation of sets. The current sets *A* and *B*, if they exist, must be deleted.

2. o — Initialize sets as ordered collections. This command allows interactive users to reset the calculator and initialize two new sets *A* and *B* using the ordered collection implementation. Existing sets *A* and *B* must be deleted.

3. e — Erase set. This command allows interactive users to delete the current A set. The previous value stored in A is lost. A is bound to a new, empty set after this command is complete.

4. s — Switch sets. The sets associated with A and B are swapped, meaning that A will receive the previous B set and vice versa.

5. c — Copy set. Set A is deep copied into B. The previous content of B is lost. The content of A is not affected. The two sets must not share any data structures, that is, they can be modified independently of each other.

6. d — Display set contents. The integers stored in the two sets are displayed on the standard output stream. The two sets are not modified.

7. a — Add element. This function allows a user to add a new integer to A. The value is obtained through an appropriate prompt with an interactive user. No action is taken if the integer in question is already in the set.

8. u — Union. This element takes the set union of A and B and stores the resulting value in A. The previous content of A is lost. Set B is not modified by this operation.

9. i — Intersection. This command takes the set intersection of A and B and stores the resulting value in A. The previous content of A is lost. Set B is not modified by this operation.

10. q—Quits the set manager.

You must work alone on this project. You are not allowed to discuss designs or share code with other students. However, you are encouraged to use the Piazza discussion board to post or answer questions about specific aspects of the project.

Save all your code in a collection of header and code files and submit a zip archive with a (short) readme file containing instructions on how to use your Set Calculator. Submit the archive by clicking on the link provided with this assignment. Your code should compile under the GNU C++ compiler. No late submissions will be accepted.

Good luck!