Due Date
Wednesday, October 16, 2013

Program objectives
The objectives of this assignment are as follows.
An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution (ABET b).

Value
This program is worth 15 points. The distribution of points will be as follows.

Criterion	Value
Global functions	1
Templates	3
Program style	3
Correct output with annotation	7
Correct insert for bag	1

Delivery method
Archive your files using the tar command (see below). Attach the tar file which will be named hw5.tar to an email that you send to the class at csc2421@orion.ucdenver.edu In the Subject field, type HW5. In the body of the email type your name, then send the mail. Archive your files as follows. Notice that all file names are in lowercase.

tar -cvf hw5.tar your_space_delimited_files_named_in_lowercase

Problem
Recall that a bag is just a multiset—a set that allows duplicates. In this project, you will create a class called **set** that is a bag that does not allow duplicates. Use this data structure to perform several set operations, that include union, intersection, and difference. Test your set operations on data read from an input file, where each line of the file consists of space-delimited integers and represents a set of integers.

Input
A data file whose name is read from argv[1], and contains 3 lines of space-delimited integers.

Output
The original sets and the sets resulting from the operations, union, intersection and difference as described below displayed on the standard output with annotations.

Program requirements
***Class* (*minimum requirements*)**
1. A default constructor that creates an empty set.
2. An insert function to add an integer to a set if possible.
3. Overloaded operators + (union), & (intersection) and – (difference) defined as follows for two sets, *A* and *B*.
   a. $A \cup B = \{x \mid x \in A \vee x \in B\}$ // Union

   b. $A \cap B = \{x \mid x \in A \wedge x \in B\}$ // Intersection
   c. $A - B = \{x \mid x \in A \wedge x \notin B\}$ // Difference

   Notice that these operations do not modify the operands.
4. Operator ==, which is true if and only if the two sets being compared have the same elements.
5. A constant member named *empty* that returns true if a set is empty and false otherwise.
6. A constant member named *size* that returns the number of elements in the set.
7. Non-member friend operator << to display the contents of a set. The operator should display a set as follows: {9, -2, 18, 4}
8. Overriden value semantics.

Driver

1. Open the program with an appropriate greeting.
2. Read the name of the data file from the command line in argv[1].
3. Open the file and read the data into 3 sets *A*, *B*, and *C*.
4. Display the contents of sets *A*, *B*, and *C* with an appropriate annotation.
5. Perform the following operations and display the results with appropriate annotations.
   a. $A \cap (B \cup C)$
   b. $(A \cup B) - C$
   c. $(A \cap B) \cup (B \cap C)$
   d. $(A - B) \cup (C - A)$
   e. $(A \cap B) == (B - C)$

   Here is an example of the output for parts 3 and 5.
   A = {2, 7, -2, 9}
   B = {12, 19, 2, 7}
   C = {-2 14, 17, 23, 19, 18, 100}
   A & (*B* + *C*) = { set elements }
   (*A* + *B*) – *C*) = { set elements }
   (*A* & *B*) + (*B* & *C*) = { set elements }
   (*A* – *B*) + (*C* – *A*) = { set elements }
   (*A* & *B*) == (*B* – *C*) = { true|false }

Notes

1. Your set must be based on the bag template. This implies that the node and linked-list toolkit must also be templates. Use the inclusion model for your template files.
2. You may combine the node and toolkit files or keep them separate.
3. The grader will create data to test your program. Do not submit a data file.
4. The overloaded operators +, &, - and == may be members or non-members.
5. The operation of 5-e above must display the literal string (either “true” or “false”).