Project 2

Set Implementation (Singly Linked List)

CS 240 - Section 1

Kristin Adachi

knadachi@cpp.edu

February 1, 2016

Section 1: Project Specification

This project emphasized the implementation of a singly linked list to create a program that utilized the standard set operations. We were required to create effective methods that would check if a set contained a certain element, add elements to a set (no duplicates), and remove a certain element if possible. We also needed to create methods that would perform the set operations union, intersection, and complement. Alongside these operations, we needed to be able to check if two sets were subsets of each other or equal to each other, regardless of whether the elements were in order or not. Finally, the user should have been able to check their current sets' elements and their sizes. I was able to meet these requirements by creating a Set class (the singly linked list with a dummy head) and a separate Node class. All of the methods listed above were implemented in the Set class. To display the sets and their sizes, I wrote a `toString()` method that traversed the set and printed its contents and size (the size was updated through the add and remove methods). I also coded a UI class that handled the user interface.

Section 2: Testing Methodology

My set program allows the user to input their own sets and a choice of operation to run on their sets. However, I tested my program mostly by using a method I hard-coded called `test()` (located in the UI class). It tested the add and remove methods, the contain method, and six special set cases. To run this default test, you need to input "8" in the main menu even though it isn't listed.

To test the add and remove methods, I tested cases such as simply adding elements to the set and removing elements from the set. Then, I tried adding numbers that were already in the set and removing numbers that didn't exist in the set. I also made sure that if all elements were removed, the resulting set would be the empty set of size zero. Secondly, I tested the contain method by adding elements and checking if the method would return true if I searched for these elements. I also checked if it would return false if I tried to find an element that didn't exist in the set.

To test the subset, equal, union, intersection, and complement methods, I used the five cases, as listed in the project description, as well as one more case. These cases included:

1. Sets A and B are equal but distinct sets.
2. Sets A and B have different sizes but one is the subset of the other.
3. Sets A and B are non-empty and different in size but have common elements.
4. Sets A and B are non-empty sets with nothing in common.
5. Set A is non-empty and Set B is empty.
6. Set B is non-empty and Set A is empty.

To check if these worked correctly, I manually wrote down each case and determined the outputs they should have produced.

Throughout my testing process, I didn't encounter many errors regarding the methods' accuracies, but I did realize that I had originally forgotten to update the size of the set in the add and remove methods.

Section 3: Lessons Learned

Throughout the process of coding this program, I became more familiar with the implementation of a singly linked list because this was the first time I had used it. Depending on its purpose, I found that using a singly linked list could potentially be more advantageous that using an array. With this particular project, it was much more convenient to use a singly linked list because the sizes of the sets weren't predetermined. I was able to add and remove elements from set without having to worry about its size. If we had needed to use an array, it would have been much less practical because we wouldn't have known how much space to allocate to each set.

Though it wasn't required, I also saw this project as an opportunity to reinforce my knowledge on creating a UI class. For the previous project (cryptography), I coded all the user interface inside the main method. Because I wanted my main method for the set program to be cleaner, I decided on making a separate class that dealt with the user interface. In the UI class, I created methods that focused on particular parts of the UI, such as the main menu and the set editor menu. I even created a method that focused primarily on editing a certain set. The only problem I faced during this process was creating the default test method. Though it is organized, I found the method to be lengthy. However, I couldn't condense it very much and left it as it is.

As a result, my main method only contained two lines as compared to the large amount of lines it had in Project 1.

In conclusion, I found that the overall designing and coding of this project wasn't too difficult. It really helped me reinforce my knowledge on previous topics while incorporating the new topic of singly linked lists. It also helped me learn how to decide whether to use a singly linked list or an array in certain situations.