CMPSC 122.2
March 22, 2022

Homework 4

Due: April 5, 2022

Overview

A set is a collection of distinct (unique) items. Using a modified version of the Doubly Linked List we discussed in class, you will implement a Set class, and provide an interactive command loop to test the Set class.

A partially implement List class is provided to you in the *list.h* file. Some of the functions need to still be defined. Once you define those functions, you will use the *list.h* file to implement your Set class. You should be able to create all of the functions of the Set class by leveraging public member functions of the List class. You do not need to use all of the List member functions. Part of the assignment is determining which functions of List are needed to create your Set class.

Program 1 (20 points) List

Finish implementing the List class (in *list.h*):

Implement a following member functions:

- *removeAt* : Remove the node at the specified index. Return the int value contained at the now removed node. Exit program if an invalid index is provided.
- *remove* : Remove the provided value if it is contained in the list. Return true if the value was found and remove, return false if no changes were made to the list.
- *at* **:** Returns the int value contained at the node at the provided index. Exit program if an invalid index is provided.
- *valueOf* : Given a ListNode pointer, return the given value
- *getNext* : Given a ListNode pointer, return a pointer to its next ListNode
- *getPrevious* : Given a ListNode pointer, return a pointer to its previous ListNode

In the *list.h* file, do not modify any of the contents of the List class unless they are marked with a **TODO** comment.

Program 2. (80 points) Set

Using the list class, implement a Set class that does the following:

1. Implements the following functions:
    - *contains* : Returns a boolean value representing if the provided int value is contained in the list.
    - *add* : Returns a boolean value representing if the provided int value was successfully added to the list. Sets are intended to contain distinct value, so do not allow duplicate values to be added. If the list already contains the supplied value, return false.
    - *remove* : Returns a boolean value representing if the provided int value was successfully removed from the list. If the value is not contained in the list, the function should return false.
    - *clear* : No return value. This function should assign 0 to set_size, and delete list.
    - *set_union* : Creates and returns a Set pointer that contains the set union of the invoking Set object and a second Set object passed into the function.
      Note: *union* is a reserved keyword in C++, which is why the identifier for this function is difference from the other two set functions.
    - *intersection* : Creates and returns a Set pointer that contains the set intersection of the invoking Set object and a second Set object passed into the function.

- *difference :* Creates and returns a Set pointer that contains the set difference between the invoking Set object and a second Set object passed into the function. The invoking Set object should be considered the set that is being subtracted from (Invoking Set – Parameter Set).
- *print* : Print the contents of the set. Use the following output format:
  `set contents(<size>) element0 element1 element2`
- A default constructor that initializes list and set_size.
- A destructor that assigns 0 to set_size and calls delete on list

2. Create an interactive "command" driven prompt system that parses "commands" from the user that are used to invoke the appropriate functions:
   - Prompt the user for the desired size of two sets, and then get the requested amount of int values and fill each set. If the user enters in a number less than 1 as the size, assume that set is empty.

   - Enter a looping prompt that parses the following commands with the appropriate parameters, and use the two sets created in the previous step to invoke the appropriate member functions:
     - *contains* <value> <set>
       - Invoke the *contains* function on the specified set, passing *value* as its parameter, print true or false based on result.
     - *add <value> <set>*
       - Invoke the *add* function on the specified set, passing *index* as its parameter, print true or false based on result.
     - *remove <value> <set>*
       - Invoke the *remove* function on the specified set, passing value as its parameter, print true or false based on the result.
     - *print <set>*
       - Invoke the *print* function on the specified set
     - *union*
       - Invoke the *set_union* function on the first set, passing the second set as its parameter. Print the contents of the resulting set.
     - *intersection*
       - Invoke the *intersection* function on the first set, passing the second set as its parameter. Print the contents of the resulting set.
     - *difference*
       - Invoke the *difference* function twice. Invoke on the first set, passing the second set as the parameter, and also invoke on the second set passing the first set as the parameter. Print the contents of both resulting sets.
     - *quit*
       - Break out of loop, exit program normally.

Expected prompt/input with example inputs:

```
Enter starting size of Set #1: 2
Enter starting size of Set #2: 4

Enter 2 values for Set #1: 1 2
Enter 4 values for Set #2: 4 3 2 1
```

```
Now accepting commands (quit to exit program):
> print 1
set elements(2): 1 2
> union
Union: 1 2 3 4
> remove 4 2
true
> print 2
set elements(3): 3 2 1
> add 5 1
true
> print 1
set elements(3): 1 2 5
> difference
First – Second: 5
Second – First: 4 3
> quit

Exiting Program.
```

Note: You must use the above format for entering the set sizes and values, as well as accepting the commands. Points will be taken off for deviating from this format.

Template Program Files

Two template files (*list.h* and *set.cpp)* will be published on Canvas.

Compiling the Program with g++

Use the following command to compile your classes. This is the command I personally use on the Sunlab machines to compile and grade your programs:

```
g++ -Wall -o <output_name> <program_name.cpp>
```

Example:

```
g++ -Wall -o set set.cpp
```

Remember: Your programs must successfully compile without any errors, or a zero will be given for that program.

Following Instructions

You are expected to follow the directives laid out in this assignment and the course syllabus. Points will be deducted for incorrectly named files, missing/incorrectly filed out banner comments, not supplying hardcopy submissions in a pocket folder with the appropriate information, and failing to implement features/functionality specified in the assignment. It is expected that you will utilize the provided template files for this assignment, and that you do not modify the names of class members/functions that are provided in the template.

CMPSC 122.2
March 22, 2022

Homework 4

Due: April 5, 2022

If you have questions about any portions of the assignment or what is expected, contact me for clarification.

Submission

- Electronic Submission (Due: One minute before midnight, 11:59 PM, Tuesday April 5 2022)

  - Your two source code files (*list.h* and *set.cpp*)
  - Submitted using CANVAS drop box