CS 247 Spring 2014
Assignment 1 Question 2 --- Specifications

Q2 [55 marks] Copy, Assignment, Equality

You are to complete the implementation of an ADT for dynamically-sized lists of strings. A `DynList` object is a list of strings whose individual string elements can be accessed using `operator[]`. The `operator[]` resizes (i.e., grows) the list whenever its index refers to an element that is beyond the current last element of the list.

Objective

We have specified the public interface for the `DynList` ADT and the data representations. You are to complete the implementation of the `DynList` ADT by providing appropriate definitions for:

- the default constructor
- the copy constructor
- the destructor
- the assignment operator
- equality operator

Your solution cannot modify the provided public interface or data representation. If you wish, you can add private member functions (e.g., helper functions) to `DynList`, and can add public member functions to `DynNode`. (`DynNode` is a private class nested within `DynList`, so any public member functions declared in `DynNode` are "public" only within the `DynList` namespace.)

Provided Code

We have provided an initial code file `DynListTestHarness.cpp` that includes the public interface for the `DynList` ADT. It also includes a partial implementation based on linked lists, including implementations of the member functions `size()` and overloaded `operator[]`. The `operator[]` resizes the list whenever its index refers to an element that is beyond the current last element of the list.

Also provided is a main program that is a **test harness,** which you can use to test your `DynList` implementation by creating lists and performing operations on them. The test harness is not robust. (It is throwaway code.) If you enter invalid commands, it might cause the program to terminate. It should go without saying (but I'll say it anyways) that your program will be tested only on valid input commands.

Commands

There are 9 valid commands that the test harness will recognize:

x <list>
d <list>
s <list>
r <list> <i>
m <list> <i> <val>
e <list1> <list2>
c <list>
a <list1> <list2> *assigns the size and contents of <list2> to <list1>*
CTL-d *terminates the program (with an EOF)*

a) x <list>

The test harness uses `DynList`'s default constructor to create a new list that can be referenced by the user-provided name <list>. The name of a <list> is restricted to values between 0 and 9 inclusive. You need to implement the default constructor method.

b) d <list>
The test harness indirectly uses `DynList`'s destructor (via a `delete` command) to destroy the specified list. You need to implement the destructor method, such that the list and its elements are deleted and any memory allocated to them is freed.

c) s <list>
The test harness calls the provided `size()` method of the specified list, and prints the value that is returned. *You do not need to do anything.*

d) r <list> <i>
m <list> <i> <val>
Command r causes the test harness to invoke the provided `operator[]` method of the specified list. It reads the <i>th element and prints its value. *You do not need to do anything*.

Command m causes the test harness to invoke the provided `operator[]` method of the specified list. It sets the <i>th element to the value <val> and prints the element's new value. *You do not need to do anything.*

e) e <list1> <list2>
The test harness uses the `DynList` `operator==` to test whether <list1> is equal to <list2> and reports the result. You need to implement the `operator==` for `DynList`.

f) c <list>
The test harness uses the `DynList` copy constructor to create a new list that is a copy of the specified list. The size of the new list must be the same as the size of <list>, and the elements of the new list must be copies of the elements of <list> and must be in the same corresponding positions. You need to implement the copy constructor for `DynList`. The test harness will print the size and contents of the copied list.

g) a <list1> <list2>
The test harness uses the DynList assignment operator `operator=` to assign the contents of <list2> to <list1>. You need to implement the assignment operator `operator=` for DynList. The test harness will print the sizes and contents of the new <list1> and the original <list2> (so that you can check that they are equal).

h) CTL-d
If you hold down the control key on your keyboard while pressing the D key, you will generate an EOF (end of file) character. When the test harness's input stream sees this, it terminates the program.

Sample Execution

Below is an example partial execution. User input is shown in **bold** font.

```
Test harness for DynList ADT:

Command: x 4

Command: x 5

Command: m 4 10 hello
Value of the 10th element of 4th list == "hello"

Command: m 4 11 world
Value of the 11th element of 4th list == "world"

Command: a 5 4
Size of 5th list = 12
Value of 5th list = [,,,,,,,,,,hello,world,]
Size of 4th list = 12
Value of 4th list = [,,,,,,,,,,hello,world,]

Command: e 4 5
Lists 4 and 5 are equal.

Command: m 4 11 foo
Value of the 11th element of 4th list == "foo"

Command: e 4 5
Lists 4 and 5 are not equal.

Command: ^D
```