cs3157 – Advanced Programming
Summer 2006, lab 5 (30 points)
June 21, 2006

Follow these step-by-step instructions. This lab must be submitted electronically (see instructions at the end of the lab) by Monday June 26, 4 pm.

Step 1. (10 points)

Download memwatch a very useful program which can make sure you aren't leaking memory.
http://www.linkdata.se/sourcecode.html
Untar ('man tar' if you need help) the file and successfully compile the program. You will need to modify it so that it outputs your name somewhere in the log.
Run the test file, which will generate a **memwatch.log** file which you will submit to show you have successfully done this step.

Compile and run the test file (see README) and document this in a short paragraph noting any issues you faced getting this to work.

Step 2 (10 points)

Adopt the String class presented in class to have the following main work:

```
int main(void)  {

String test;     //default constructor triggered here
String test2;
test = "Hello!";   //= operator
test2 = "Bye";
String test3 = test + test2;
String *str = ???????    //pointer at test3
if( test != test2){
cout << "the string are not equal" << "String 1 is " << test << "
String 2 is " << test2 << endl;
}else if(test2 == ???????){  //compare to str pointer
cout << "test3 is " << test3 << endl;
}


return 0;
}
```

Don't forget to clean up your code and add plenty of comments. Also replace the ???? with the correct stuff.

Step 3 (10 points)

The vector class in java allows you to create an array which grows as needed. We will be implementing the same in cpp but a little differently.

Create a base class called List, which defines functions

```
void add(double)        //this will add the double into our list
void get(int)           //return the nth element, remember for negative numbers
                        //to return the list offset from the end like perl does.
void delete (int)       //delete the int-th spot
Int getSize()           //returns current size
void append(List l2)    //append the contents of L2 to end of current list
```

Now derive a FloatList which derives from base and adds the following:

```
double average()        //returns the average
double stddev()         //returns the list's standard deviation
```

Now derive a string list which will store a string list. You want to make sure that for a given sequence of add and delete you will not be leaking memory.

Create an appropriate main to test all functions for all classes. Include some pointer tests too.

Submit your lab.