CS 132 - Intro to Computer Science II - Spring 2005
Week #3 Lab Exercise and Homework #2

Week #3 Lab Exercise due: Wednesday, February 2nd, END of lab
HW #2 due: Wednesday, February 9th, beginning of lab

Opening comment block requirements (for the homework and the lab exercise):

* these are the same as for HW #1, **except** that "`avg case time complexity`" is **NOT** required; see the posted "opening comment block for a function" in the posted templates.
* header files and test main's only require the opening comment block formats described for such files in HW #1 (and also included in the posted templates).
* functions require all of the "pieces" seen in binSearch.cpp except for the avg case time complexity. Further examples of such opening comment blocks for functions can be seen in the posted examples (see "In-class Examples" from the course web page) from Lecture 3: fib.cpp, binSearch.cpp, writeBackward.cpp, and recMax.cpp.

Other style reminders:

* remember to use **EXIT_SUCCESS** for returns from a main function.
* you are expected to use **assert** to ensure preconditions when it is reasonable to do so.
* you are expected to choose representative examples for all non-main functions; for recursive functions, note that this includes testing all base cases as well as main categories of recursive cases.
* you are expected to use **#ifndef** appropriately in .h (header) files (as is demonstrated in the posted template for header (.h) files).

Week #3 Lab Exercise

You m ay work individually or in a pair for this lab exercise. IF you work together, you must include BOTH of your names in the resulting code's opening comment blocks, and both must contribute to all of the work done in the assignment. You are responsible for understanding all of the code that is turned in with your name on it.

(adapted from Carrano and Prichard, "Data Abstraction and Problem Solving with C++", p. 104) Given an integer **n > 0**, write a ***recursive*** C++ function **writeList** (in file **writeList.cpp**, with header in file **writeList.h**) that writes the integers 1 2 ... **n** to the screen as a blank-separated list (with no newline at the end). The last item may be followed by a blank --- that is, if you were to write:

```
cout << "<";
writeList(13);
cout << ">" << endl;
```

you'd see:

```
<1 2 3 4 5 6 7 8 9 10 11 12 13 >
```

(Yes, it'll just spew out a *reallllly* long line if you call it with a large **n** --- that's okay.)

Write a main function in file **test_writeList.cpp** to test **writeList** using the examples from its opening comment block in the same style as **test_writeBackward.cpp** from the posted Lecture #3 examples (since it is a **void** function like **writeBackward** is).

Compile, test, run, debug, repeat as necessary --- when you are satisfied with your code, put your name on the Next: list on the board, so that your files can be checked.

To receive credit for this lab exercise, the above must be completed by the end of the lab period.

Homework #2

You are to work individually on these homework problems.

1. (adapted from Carrano and Prichard, "Data Abstraction and Problem Solving with C++", p. 104) Write a ***recursive*** function **recSum** (in file **recSum.cpp**, with header in file **recSum.h**) that will compute the sum of the first **n** integers (**n** >= 0) in an array of at least **n** integers. (Hint: begin with the **n**th integer...) You are expected to assume that the sum of zero integers (**n==**0) should be 0.

Write a main function in file **test_recSum.cpp** to test **recSum** using the examples from its opening comment block in the same style as **test_binSearch.cpp** and **test_seqlSearch.cpp** from HW #1 and **test_recMax.cpp** from the posted Lecture #3 examples.

Compile, test, run, debug, repeat as necessary --- when you are satisfied with your code, run:

```
test_recSum > prob2_1_out
```

...to create an example output to turn in. You only need to turn in **recSum.cpp**, **test_recSum.cpp**, and **prob2_1_out**, however.

2. (adapted from Carrano and Prichard, "Data Abstraction and Problem Solving with C++", p. 104) Write a ***recursive*** C++ function **reverseNum** (in file **reverseNum.cpp**, with header in file **reverseNum.h**) that writes the digits of a positive integer in reverse order to the screen. (Like in-class example **writeBackward**, it should not add a newline to the end. Hint: this is a variation on the example **write_vertical** from Ch. 9 of the course text.)

Write a main function in file **test_reverseNum.cpp** to test **reverseNum** using the examples from its opening comment block in the same style as **test_writeBackward.cpp** from the posted Lecture #3 examples (since it is a **void** function like **writeBackward** is). (Note that < >'s should not be necessary in this particular case, either, although they "work" better here than they would in writeList from the lab exercise above.)

Compile, test, run, debug, repeat as necessary --- when you are satisfied with your code, run:

```
test_reverseNum > prob2_2_out
```

...to create an example output to turn in. You only need to turn in **reverseNum.cpp**, **test_reverseNum.cpp**, and **prob2_2_out**, however.

3. (adapted from Carrano and Prichard, "Data Abstraction and Problem Solving with C++", p. 105) Write a ***recursive*** C++ function **writeLine** (in file **writeLine.cpp**, with header in file **writeLine.h**) that writes a character repeatedly to form a line of **n** characters followed by a newline to the screen. For example, `writeLine('*', 5)` produces the line

```
*****
```

Note that **n==0** should cause just a newline to the written to the screen (no characters then a

newline).

Write a main function in file **test_writeLine.cpp** to test **writeLine** using the examples from its opening comment block in the same style as **test_writeBackward.cpp** from the posted Lecture #3 examples (since it is a **void** function like **writeBackward** is). (Note that < >'s may be problematic in this case, because of the newline.)

Compile, test, run, debug, repeat as necessary --- when you are satisfied with your code, run:

```
test_writeLine > prob2_3_out
```

...to create an example output to turn in. You only need to turn in **writeLine.cpp**, **test_writeLine.cpp**, and **prob2_3_out**, however.

4. (adapted from Carrano and Prichard, "Data Abstraction and Problem Solving with C++", p. 105) Now write a ***recursive*** function **writeBlock** (in file **writeBlock.cpp**, with header in file **writeBlock.h**) that **uses** Problem #3's **writeLine** to write **m** lines of **n** characters each to the screen. For example, `writeBlock('*', 5, 3)` produces the output

```
*****
*****
*****
```

Write a main function in file **test_writeBlock.cpp** to test **writeBlock** using the examples from its opening comment block in the same style as **test_writeBackward.cpp** from the posted Lecture #3 examples (since it is a **void** function like **writeBackward** is). (< >'s are probably problematic here, too.)

Compile, test, run, debug, repeat as necessary --- when you are satisfied with your code, run:

```
test_writeBlock > prob2_4_out
```

...to create an example output to turn in. You only need to turn in **writeBlock.cpp**, **test_writeBlock.cpp**, and **prob2_4_out**, however.

Make sure, when you are done, that you have submitted to me the following files (using **~st10/132submit** on cs-server):

recSum.cpp, test_recSum.cpp, prob2_1_out
reverseNum.cpp, test_reverseNum.cpp, prob2_2_out
writeLine.cpp, **test_writeLine.cpp**, **prob2_3_out**
writeBlock.cpp, **test_writeBlock.cpp**, **prob2_4_out**