

CISC181H Spring 2009 Lab04

- Write a program for each of the following problems. Be sure to save every separate program. All programs must be properly commented and indented (see Assignment Standards on the class website). Ask your TA for guidance.
- Name each program lab02.n.cc, where n is the number in the list below. For example, the name of the file for the first will be lab02.1.cc. Put the files in your lab02 directory (see Anderson for instructions on how to make a directory).
- Your book is an excellent resource. I find the table of contents very good, and the index not so good. Try looking in the TOC for topics you want reinforced.
- Do not be afraid to write multiple functions for a single problem. Short, simple functions are much more likely to work, and tend to be easier to write.
- I program a few lines of code at a time (sometimes just one!) then I compile and test. This saves much effort later. If a bug occurs, I know right where to look. Usually the easiest lines to type in and test are from the middle of a problem, not the beginning.

Programs

1. Review the first three problems of lab03. For each, make a drawing that shows what memory is on the stack, what is on the heap, what is contiguous vs. not, and where those extra two chars go.
2. Write a program with two integer arrays and one double array of size 10. Use the rand() function¹ to place integer values in the first array; the values should be between 1 and 100, inclusive².

In the second array, place the running sum of the first array. In the double array put the running average.

Example:

```
> runningAvg
8      50      74      59      31      73      45      79      24      10
8      58      132     191     222     295     340     419     443     453
8      29      44      47.75   44.4   49.1667 48.5714 52.375  49.2222 45.3
```

Finish the above first. Then copy the files runningSumAvg.o and runningSumAvg.h from the lab04 directory using the Unix cp command (do not cut and paste). In the .h file you'll find the prototype for a test function. The definition of the test function is in the .o file, already compiled into an "object" file.

To use the test function, at the top of your .cc file add the line³:

```
#include "runningSumAvg.h"
```

¹You may need to #include the cstdlib file to use rand().

²Accomplish this using the mod operator for scaling, as in your text

³What will this line do?

and then add a call to the test function in main with your arrays as parameters⁴. Then compile using

```
g++ runningSumAvg.o yourFileNameHere.cc
```

and demonstrate your program running and printing.

3. **Do not do this problem** until you have completed problem 2. Compile bug.cc with the object file runningSumAvg.o, as shown above. Run the resulting executable and observe the error report.

Now write your own version of the test file runningSumAvg.cc. Give it a more useful error message than the one I wrote. Demonstrate it working on the program in this directory, bug.cc. You may need to be working on Strauss for bug.cc to work “correctly”, i.e. if the test does not fail then move to Strauss where it will⁵.

Explain in writing exactly what the error is in bug.cc. What causes the error? How does it manifest? **SHOW** the result of the error as part of your explanation.

4. Write a makefile that separately compiles two .cc files to object code, then separately compiles the object files to executable when I type “make”. If I type “make test” it should check to see that the executable is up to date, then run it⁶.
5. Create an integer array of size 10. Read ten data points into the array from a file, and use a function to print the array nicely. Then modify the contents of the same array so that every location contains the sum of all numbers to the left and itself. So if the inputs are 2, 5, 3, ..., the new contents would be 2, 7, 10, Write the new contents to a new file.
6. Read an entire file of integers (100 or more). Your program will start with an array of size one, double the size when it is full, and then continue reading. Print each full array in a nice way that does not eat paper.
7. Same as last problem, but read words.

You should have a total of 7 programs named lab04.1.cc to lab04.7.cc, plus your game code and game test code. Make a single script file (see lab00 for the instructions) where you cat, compile, and run lab code in its final form. For the game, the TA will run your submitted testing and game code during the next lab.

Submit all 7 program files *and* your script on WebCT by midnight before your next lab. Give the paper version of the complete script file **only** on paper to your TA at the **beginning** of your next lab. Note: Cat, compile, and run each program in order - do *not* cat all programs, then compile, etc.

⁴If you had to write code for a file like runningSumAvg.o what would it look like?

⁵or at least, it always has for me.

⁶Normally this would be a different executable than the one for “make”, but we can pretend.