Homework 1 (100 points)

Data Structures and Algorithms in C++
Shlomo Hershkop
Department of Computer Science
Columbia University
Summer 2006

Out: Wednesday July 5
Due: Tuesday July 11 11pm (Electronically, see webpage for submission instructions).

Goal: review recursion, running time, analysis, linked lists.

1. Setup any graphical c++ environment. Try to compile the Fig01_06.cpp file off the source code from the book (see tools section on courseworks).
   Write-up a short paragraph on your environment of your choice and document any issues you might have had getting things to work. Please include your working system (cunix, windows, etc)

2. Bill and Mary are arguing about the performance of their sorting algorithms. Bill claims that his O(n log n)-time is always faster than Mary's O(n^2)-time algorithm. To settle the issue, they implement and run the two algorithms on many randomly generated data sets. To Bill's dismay, they find that if n < 100 the O(n^2)-time algorithm actually runs faster, and only when n >=100 is the O(nlogn)-time one better. Explain why this scenario is possible. You may give numerical examples to illustrate your point.

3. Here is the recursive factorial function:

```
public static int factorial (int n)
{
  if (n <= 1)
    return 1;
  else
    return n * factorial (n-1);
}
```

   I claim that all recursive programs can be written in a non-recursive fashion. Rewrite this function so that it does not use recursion.

   Make sure your program compiles and works correctly and include a short main with assert to check that it really works.

4. Write a recursive method in C++ pseudo code that returns the number of 1s in the binary representation of N. Feel free to lookup how C++ handles bit with bit manipulation operators or come up with your own approach. You must solve the problem in a recursive fashion.

   example: 23 = 10111 would return 4

   28 = 11100 would return 3

Hint: Use the fact that this is equal to the number of 1s in the representation of N/2, plus 1 if N is odd.

5. Prove, using mathematical induction, that the sum for i = 1 to n of i^2 is $(2n^3 + 3n^2 + n) / 6$.

6. Prove, using mathematical induction, that the sum for i = 1 to n of $1/(2^i)$ is $1 - (1/(2^n))$.

7. Arrange the following expressions by growth rate from slowest to fastest: $14n^2$, $\log_3 n$, 3^n, $20n$, 2, $\log_2 n$, $n^{2/3}$, $n \log_2 n$, $n!$.

8. Suppose that a particular algorithm has time complexity $O(n^2)$, and that executing an implementation of it on a particular machine takes T seconds for n inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in T seconds?

9. Give an analysis of the Big-Oh running time for each of the following program fragments:

```
sum = 0;
for (i = 0; i < 3; i++)
  for (j = 0; j < n; j++)
    sum++;
```

```
sum = 0;
for (i = 0; i < n*n; i++)
  sum++;
```

Assume the array contains n values.

```
for (i = 0; i < n; i++) {
  for (j = 0; j < n; j++)
    array[i] = random(n);    // random() takes constant time
  sort(A, n);                // sort takes n log n time
}
sum = 0;
if (EVEN(n))                 // EVEN(n) is true iff n is even
  for (i = 0; i < n; i++)
    sum++;
else
  sum = sum + n;
```

10. Write and explain how a series of C++ statements would use the LinkedList class to delete every third element in a given list. (for example item 3, 6, 9 etc).

11.
   a) When I want to send you a message M over the Internet, I can breaks M into n data packets, numbers the packets in order, and inject them into the network. When the packets arrive at your computer, they may be out of order, so you must assemble the sequence of n packets in order before you can be sure you have the entire message M. describe an efficient scheme for you to do this.
   b) What is the running time of your algorithm?

Programming Section : This is a very straightford programming assignment, helping you get started with C++ again and getting comfortable with the C++ IDE.

You will be graded on a few important points including:

1. Clear code
2. A readme file explaining all the files included and what each one does. Also instructions on how to run each step
3. plenty of comments, will help step 1
4. clear variable names, a good IDE will allow you to automatically type out longer (but not ridiculous) variable names.
5. No memory leaks, remember for each new you need a delete.

Step 1) Download the http://www.cs.fiu.edu/~weiss/dsaa_c++3/code/List.h source code. Write a small main to test randomly inserting 30 integers between 0-100 (can have repeats) into the list and printing them out.

You should provide a makefile to compile and run step1.

I will provide a sample makefile online. In addition see http://www.eng.hawaii.edu/Tutor/Make/ for a tutorial.

2) Now adopt the insert method and the class code so that a counter is kept with each item in the list. When insert happens, and the item is already present you should increment the counter instead of inserting it again in the list. So inserting 2,1,3,4,3 will only have a list of 2,1,3,4 but 3's count will be 2.

3) Now insert 100 random numbers between 0-20. Adopt a print method so that you can print the list, and output each item along with a frequency count :

```
2 14.29%
1 28.57%
3 14.29%
```

etc