Test 2 will be Thursday November 1st in class. It will be closed-book and notes, except for one 8.5” x 11” sheet of paper containing any notes that you want. (Yes, you can use both the front and back of this piece of paper.) Plus, you can use your Python Summary handout.

The test will cover Chapters 4 and 5. The following topics (and maybe more) will be covered:

**Chapter 4: Recursion**
Recursive functions: base-case(s), recursive case(s), tracing recursion via run-time stack or recursion tree, “infinite recursion”
Costs and benefits of recursion
Recursive examples: countDown, OrderedList __str__ method, fibonacci, factorial, binomial coefficient
Divide-and-Conquer technique of solving a problem. Examples: fibonacci, coin-change problem
Backtracking technique of solving a problem: Examples: coin-change problem, maze (textbook)
General concept of dynamic programming solutions for recursive problems that repeatedly solve the same smaller problems over and over. Example fibonacci, coin-change problem, binomial coefficient

**Chapter 5: Searching and Sorting**
Sequential/Linear search: code and big-oh analysis
Binary Search: code and big-oh analysis
Python List implementation (ListDict) of dictionaries and big-oh analysis
Hashing terminology: hash function, hash table, collision, load factor, chaining/closed-address/external chaining, open-address with some rehashing strategy: linear probing, quadratic probing, primary and secondary clustering
Hashing implementation of dictionaries (ChainingDict and OpenAddrHashDict) and their big-oh analysis
General idea of simple sorts
Simple sorts: selection, bubble, insertion sorts and their big-oh analysis
Advanced sorts and their big-oh analysis: heap sort, quick sort and merge sort
Mergesort

1. Create empty heap
2. Add all n items
3. Dequeue back to sorted array

Mergesort

Unsorted n

Unsorted n/2

Sorted n/2

Sorted n/2

Merge
Worst case

\[
\begin{align*}
\frac{n}{2} \geq \text{pivot} \quad \frac{n}{2} < \text{pivot}
\end{align*}
\]

\[
O(n \log n)
\]

\[
\frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}
\]

\[
\text{comparsions}
\]

\[
\text{moves}
\]
Data Structures - Test 2

Question 1. (10 points) What is printed by the following program? Output:

```python
def recFn(a, b):
    if a < 0:
        return 100
    elif b < 0:
        return 1000
    elif a > b:
        return recFn(a - 3, b - 5)
    else:
        return recFn(a - 1, b - 3) - b
    print("Result = ", recFn(8, 10))

Output:
8 10
7 7
6 4
3 1
Result = 989
```

Question 2. a) (12 points) Write a recursive Python function to compute the binomial coefficient using the following recursive definition of $C(n, k)$:

\[ C(n, k) = C(n-1, k-1) + C(n-1, k) \]

for $1 \leq k \leq (n-1)$, and

\[ C(n, k) = 1 \]

for $k = 0$ or $k = n$

```python
def C(n,k):
    if k==0 or k==n:
        return 1
    else:
        return C(n-1, k-1) + C(n-1, k)
```

b) (8 points) For the above recursive function $C(n,k)$, complete the calling-tree for $C(4,2)$.

c) (3 points) What is the value of $C(4,2)$? 6

d) (2 points) What is the maximum number of call-frames of $C$ on the run-time stack when calculating $C(4,2)$ recursively? 4