COSC 350: Quiz 2
Fall 2008: Sections 2.7, 4.3, & 4.4

Be sure to answer each question carefully and completely.

1. (1 pts) Enter your name: _____

2. (12 pts.) Using the hash function *h(n) = n % 10* with *collision resolution* by *linear probing*, insert the following numbers into the given hash table:

   277, 63, 500, 127, 89, 958.

3. (3 pts.) What is the *loading factor* for the previous problem?

4.(3 pts.) What is the ideal complexity for hashing?

5. (3 pts.) What problem with *linear probing* is avoided with *chaining*?

6. (4 pts.) What are the ideal characteristics of a "perfect" hash function?

7.(10 pts.) Which would you prefer to use? Both? Neither?

Conditions	Choices	Answer
The algorithm must be stable	Insertion vs. selection sort	
The system does not support recursion	Quick vs. merge sort	
The data is almost completely sorted	Bubble vs. insertion sort	
Data must be sorted in place	Merge vs. bubble sort	
You need the lowest possible average complexity	Insertion vs. shell sort	

8. (6 pts.) Complete the table:

Algorithm	Best Case Complexity	Average Complexity	Worst Case Complexity
Binary Search			
Linear Search			

9. (18 pts.) For the given data, show the steps the data will go through with the given sorting algorithms. Give the data after each swap or block swap.

Bubble Sort					
D	A	B	F	C	E

Insertion Sort					
D	B	G	A	F	E

Selection Sort					
D	B	G	A	F	E

10. (8 pts.) Give Python code to define a *node* class that might be used in a *linked list* implementation similar to the one in Section 7.2 of your text.

11. (12 pts.) Give Python code for a *partial* class definition of a *linked list* similar to the one in Section 7.2 of your text. Use the *node* definition given in the preceding problem. Your definition should include members to

    a) initialize a linked list
    b) find the size of a linked list
    c) create and add a new node to the front of the linked list

12. This is a continuation of the previous problems. Assume you have a non-empty linked list **`mylist`** and a reference (or pointer) to a node **`current`**. You want to add a new node with the stored value **`datum`** so that the new node follows **`current`**, i.e., the new node is spliced into the list after the **`current`** node.

    a) (4 pts.) What two cases must you consider when designing the code?

    b) (4 pts.) Use *box-pointer* notation to sketch each case both before and after the addition.

    c) (12 pts.) Give Python code that will work for both cases.