Practice Problems.

Complexity.

1. Consider a small array a and large array b. Accessing the first element of a takes more/less/the same amount of time
as accessing the first element of b.
    A. the same amount of time
    B. less time
    C. more time
2. True/False? Big-O is used as an upper bound
   a) $2^{N/2} = O(2^N)$
   b) $N^2 + N^2 = O(N^3)$
   c) $100\ N^3 = \Omega(N^2)$
   d) $N \log N + N^2 = \Theta(N^2)$

Also prove your answer for each question above

3. What is the complexity of the mergeStep() method (merges two arrays into one)?

4. Big-Theta complexity?

```
for (int j = 1 ; j < n ; j *= 2)
   for (int I = 1; i<n; i++)
      {some_statement;}
```

Generics.

1. Does this compile? If not, why not? And how do you fix it?

```
public static void myMethod( Collection<List> arg ) { ... }
```

// in main

```
Collection<LinkedList> myL = new Collection<LinkedList>();
myMethod( myL );
```

2. Which of the following compile?

- Collection<List> c = new LinkedList<List>();
- LinkedList<List> myL = new LinkedList<List>();
- LinkedList<List> myL = new LinkedList<ArrayList>();
- LinkedList<? extends List> myL = new LinkedList<ArrayList>();
- LinkedList<? super List> myL = new LinkedList<List>();
- LinkedList<? super List> myL = new LinkedList<Collection>();

- LinkedList<Collection> myL = new LinkedList<Collection>();
  myL.add(new ArrayList());

3. What are generic classes? Why do we use them? Clearly explain the advantage(s). Give an example generic class from the standard Java collection library.

C questions.

1. Consider:

```
int a[3];
int *p = a;
```

The value stored at the memory address found in p is:

A: the name a
B: the address of the first slot of the array
C: the value of the first element of the array
D: undefined, since the operation is illegal

2. The array access a[3] can be rewritten as:
   A. *(a) + 3
   B. *(a) + 2
   C. *(a + 3)
   D . *(a + 2)

3. Consider:
   int a[] = { 10, 100, 100 };
   int *p = a;
   The expression p + 1:

A: points to the second slot of a
B: evaluates to 101
C: points to the first slot of a
D: evaluates to 11

4. The string "dog" is stored as:

A: three non-contiguous memory locations
B: 4 contiguous locations in memory
C: a single location in memory
D: 3 contiguous locations in memory

Heaps.

1. Binary Min Heaps. Recall that a binary min heap with n elements can be stored in an array A, where A[1] contains the root of the tree. Given that values are stored in A[1] to A[size], BubbleUp and TrickleDown are defined below:

```
// Move the value A[i] up as needed.
void BubbleUp (int[] A, int size, int i);

// Move the value A[i] down as needed.
void TrickleDown (int[] A, int size, int i);
```

a) Implement the `deleteMin` method (as described in class) below. You may call `BubbleUp` and `TrickleDown` as needed:

```
// Given that values are stored in A[1] to A[size],
// remove and return the smallest value in the heap.
int deleteMin(int[] A, int size) {
```

b) Implement the `BubbleUp` method below:

```
// Given that values are stored in A[1] to A[size],
// move the value A[i] up as needed.

void BubbleUp(int[] A, int size, int i) {
```

2. Give one reason why inserting a value into a d-heap containing N items might be faster than inserting a value into a binary heap of the same size.

3. Draw the binary min heap that results from inserting 12, 1, 3, 7, 4, 5, 15, 0, 6 in that order into an initially empty binary heap.

4. Draw the result of doing 2 deletemins on the heap you created in problem 3.

5. Give the worst possible case estimate for finding the maximum of a binary min heap. Explain.

BST.

1. Draw the BST that results from inserting 12, 1, 3, 7, 4, 5, 15, 0, 6 in that order into an initially empty tree.

2. Remove the root from the tree you just created.

3. What is the postorder, preorder and inorder traversals for the tree from problem 2?

4. How many edges are in a full binary tree of height h?

5. Given the following orders of traversals of a binary tree, recreate the tree:
Postorder: G, D, B, H, I, E, F, C, A
Inorder: D, G, B, A, H, E, I, C, F

Hash Tables

1. Consider a set of objects with hash codes 40, 5, 18, 29, 35, 7 and a hash table of size 11.
a) Hash the above objects using separate chaining.
b) Liner Probing
c) Quadratic Probing

2. What is the best definition of a collision in a hash table?

A. Two entries are identical except for their keys.
B. Two entries with different data have the exact same key.
C. Two entries with different keys have the same exact hash value.
D. Two entries with the exact same key have different hash values.

3. Convert word "coffee" into an integer (hash table index) using Horner's method. (Use base = 32).

4. Someone wants to store an **ordered** dictionary in a hash table. Is it a good idea? Explain.

Coding questions

1. 8Write a method to implement a merge sort (including the method for merge step).
2. Given two strings, check if they're anagrams or not. Give the most efficient algorithm possible. (Hint: it should be O(n)).
3. Recursion (if you need more practice go here:
http://codingbat.com/java/Recursion-1
http://codingbat.com/java/Recursion-2

4. Write a recursive Java method `int sameStack(StackLi s1, StackLi s2)` to test whether two integer stacks contain the same elements. The method will return 1 if the stacks contain the same elements and 0

otherwise. You can use the methods below in your method, and don't worry about how the data type is implemented.

```
int pop()
void push(int x)
boolean isEmpty()
```

Sorting

1. What needs to be true about the choice of pivot value to ensure that quicksort has an expected time of O(n log n)?

2. A newspaper route has recently been computerized. Information about each of the 100 customers is stored in individual records containing first name, last name, and payment due. In writing a computer program to process the customer records, the programmer is uncertain whether to add a procedure to sort the records.
   1) If the records are not sorted, what is the maximum possible number of comparisons that must be made to obtain a particular customer's record using a sequential search?
   A. 100, B. 50, C. 7, D. 5 E. 3
   2). If the records are first sorted, what will be the maximum number of comparisons needed with a binary search to find a particular customer's record?
   A. 100 , B. 50, C. 7, D. 5, E. 3

3. In a selectionsort of n elements, how many times is the swap function called in the complete execution of the algorithm?
   A. 1
   B. n - 1
   C. n log n
   D. n^2

4. Selectionsort and quicksort both fall into the same category of sorting algorithms. What is this category?
   A. O(n log n) sorts
   B. Divide-and-conquer sorts (recursive)
   C. Comparison sorts
   D. Average time is quadratic.

5. Suppose we are sorting an array of ten integers using a some quadratic sorting algorithm. After four iterations of the algorithm's main loop, the array elements are ordered as shown here:

1 2 3 4 5 0 6 7 8 9

Which statement is correct? (Note: Our selectionsort picks largest items first.)

A. The algorithm might be either selectionsort or insertionsort.
B. The algorithm might be selectionsort, but could not be insertionsort.
C. The algorithm might be insertionsort, but could not be selectionsort.
D. The algorithm is neither selectionsort nor insertionsort.

6. 10. What is the worst-case time for quicksort to sort an array of n elements?

A. O(log n)
B. O(n)
C. O(n log n)
D. $O(n^2)$

7. Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2 5 1 7 9 12 11 10

Which statement is correct?

A. The pivot could be either the 7 or the 9.
B. The pivot could be the 7, but it is not the 9.
C. The pivot is not the 7, but it could be the 9.
D. Neither the 7 nor the 9 is the pivot.

8. Which of the following sorting algorithms perform in O(n) time in the best case?

a. Insertion Sort
b. Bubble Sort
c. Selection Sort
d. All of the above
e. None of the above

Missalenous

1) Of the following data structures, which would be best suited for an application where search time must be minimized and where data can be easily added?

A. Sorted array
B. Hash table
C. Sequential file
D. Circularly linked list with two header nodes
E. Singly linked list

2) What does the acronym “ADT” stands for? What is an ADT? What is the connection, if any, between an ADT and a data structure? Use an example to illustrate the answer to your last question.

3) Why does a queue implementation using a linked list require a tail pointer but a stack implementation does not?

4) What is the best data structure to solve the following problem? A list needs to be built dynamically. Data must be easy to find, preferably in O(1). The user does not care about any order statistics such as finding max or min or median.
First assume that the index is given. Then assume that it is not given.

i. Use an Array
ii. Use a Singly LL
iii. Use a Stack
iv. Use a Queue
v. None of the above