Review Exam 1

---

(1) Java
  —primitive types, loops, conditionals
  —instance variables, parameters, local variables, static variables
  —scope of variables
  —classes and objects; methods, constructors
  —this
  —methods inherited from Object
  —inheritance, interfaces
  —type casting

(2) Sorting and searching
  —linear and binary search
  —bubble sort, insertion sort, selection sort

(3) Linked lists
  —lists vs arrays
  —operations on lists
  —singly LL, doubly LL

(4) Program analysis
  —growth rate: big-Oh, big-Theta
  —finding the order of growth of an expression
  —analyzing running times of algorithms
  —comparing algorithms

(5) Recursion
  —simple recursion examples
  —towers of hanoi
  —blob counting, flow
  —generating permutations, subsets, subset sum

(6) Stacks and queues
  —stacks and queues as generic data structures (i.e. without pinning down an implementation)
  —queues with arrays vs linked lists
  —stacks with arrays vs linked lists

The test will have two parts. One part will be in-class, closed books, pen-and-paper, about one hour. The second part will be programming. You can take it home, and it will be due a few days after the exam.

A good way to study is to try to solve the problems thet we went over in class but without looking at the lecture notes (just pretend you are taking the test and seeif you can come up with the solutions on your own).

Practice problems

(1) Describe the concept of *abstraction* and how you implement it in your code.

(2) Write a method `filterString` that takes two strings as input, `str` and `filter` and removes all occurences of all characters in `filter` from `str`.

```
String filterString(String str, String filter)
```

Example: filterString("Bowdoin", "bod") should return "Bwin".
What is the runnig time of your algorithm, in terms of n_1 and n_2, the lengths of `str` and `filter`?

(3) [from Nick Parlante, Stanford] Write a method `removeDuplicates()` which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally the list should only be traversed once.

```
void removeDuplicates(LNode head)
```

(4) [from Nick Parlante, Stanford] Write an interative method `reverseList()` that reverses a list by rearranging all the pointers and head pointer.

```
void removeDuplicates(LNode head)
```

(5) [from Stanford, CS106B 2008] Determine the computational complexity (use Theta notation) of the following methods and briefly justify your answer.

```
int Mystery1(int n) {
   int sum =0;
   for (int i=0; i<n); i++) {
      for (int j=0; j<n; j++) {
           sum+= i*j;
      }
   }
   return sum;
}

int Mystery2( int n) {
   int sum=0;
    for (int i=0; i<100); i++) {
      for (int j=0; j<n; j++) {
           sum+= i*j;
      }
   }
   return sum;
}


int Mystery3( int n) {
```

```
    if (n<= 1) return 1;
    return Mystery3(n/2) + 1;
}
```

(6) [from Stanford, CS106B 2008] Yuo have an array containing n items currently unsortd. You are trying to decide whether it is worthwhile to sort the data before performing repeated searches, to take advantage of binary search. If you use selection sort to sort the data, how many searches do you need to perform to "buy back" the cost that went into sorting? Express the answr as a function of n. Evaluate your answer (do not use a calculator) for
(a) $n = 16$
(b) $n = 1024$

(7) The number of operations executed by Algorithms A and B is $8n \lg n$ and $2n^2$, respectively. Determine n_0 such that A is better than B for all $n \geq n_0$.

(8) Order the following functions by asymptotic growth rate:

$$4n \lg n + 2n, 2^{10}, 2^{\lg n}, 3n + 100lgn, 4n, 2^n, n^2 + 10n, n^3, n \lg n$$

(9) Suppose that you are given an n-element array A containing distinct integers that are listed in increasing order. Given a number k, describe a recursive algorithm to find two integers in A that sum to k, if such a pair exists. What is the running time of your algorithm?

(10) [from Stanford, CS106B 2008] Review the definition of the Big-Oh notation. We use this notation to express the *asymptptic* running time of an algorithm as the input size grows. But sometimes, due to the constant factors, a function with a larger Big-Oh can outperform a function with a smaller Big-Oh.
So you are trying to chose whether to use selection sort or merge sort to sort some amount of data. As we saw in class, selection sort has a running time of $O(n^2)$ while merge sort has a running time of $O(n \lg n)$. However merge sort has larger constant factors than selection sort and for some inputs these can outweigh the advantage of being $n \lg n$ rather than n^2. Assume selection sort has a constant factor of 10, and assume merge sort has constant factor of 100.

(a) Which algorithm would you chose if you needed to sort 50 items?

(b) Which algorithm would you chose if you needed to sort 100 items?

Based on the above, if you were writing a sorting function (and could only use selection sort or merge sort or a hybrid approach) what would you do to make it as fast as possible?

(11) Write a recursive function to print an integer in binary:

```
void printBinary(int n)
```

Examples:
$n = 0 ==> 0$
$n = 1 ==> 1$
$n = 2 ==> 10$
$n = 3 ==> 11$
$n = 4 ==> 100$

(12) Using the algorithm studied in class, draw the recursion tree for generating permutations for the input "abcd".

(13) Using the algorithm studied in class, draw the recursion tree for generating all subsets for the input "abcd".

(14) Write a method that generates all the anagrams of a given word. Use the method `generatePermutations(String s)` as a building block; assume your method takes as parameter a lexicon of valid words, and assume that you have a function that searches the lexicon efficiently.
What is the runing time of your algorithm in the worst case, as a function of n (in this case n is the size of the input string)?

(15) I encourage you to check out the site maintained by Nick Parlante with (free) online practice problems: `http://javabat.com/`. Try the recursion problems!