CS 145 Midterm #2 Practice Problems

(1) **Lists, Maps, and Sets**

Write a method that accepts a List of Strings, and a Set of Characters and determines the number of times that each character is found in a word as a map. Do not count doubles.

For Example the set ('b','o') and the List ("book", "love", "born"). Would return a set that maps ['b' = 2 : 'o' = 3]

(2) **Sorting**

By hand showing each loop through the algorithm.

sort the array **[h, y, e, r, u, a, z, i]**

Using:

- Selection
- Insertion
- Bubble
- Merge Sort

(3) **Linked Lists**

Write a method **pivot()** that can be added to a linked list object that would swap every other element around. If there is an odd number, then leave the last element alone.

```
[2, 3, 4, 5, 6, 7]  pivot() ->  [3,2, 5, 4, 7, 6]
[1, 5, 6, 2, 1, 2, 5]  pivot() ->  [5, 1, 2, 6, 2, 1, 5]
[1, 2]  pivot() ->  [2, 1]
[3] pivot() -> [3]
```

You are not allowed to construct any new nodes to solve this problem and you are not allowed to change any of the integer values stored in the nodes. You must solve the problem by rearranging the links of the list.

(4) **Stacks and Queues**

Imagine that you have a queue called x and a stack called y. What would be the result and final values of each data structure given the following cdoe?

```
1   x.add(100);
2   x.add(200);
3   y.push(300);
4   y.push(400);
5   x.add(y.pop() );
6   y.add(x.remove() );
7   System.out.println(x.peek()  );
8   System.out.println(y.peek()  );
9   x.add(y.pop() +   y.pop());
10  x.add(500);
11  x.add(600);
12  y.push(700);
13  y.push(800);
14 System.out.println(x.removr()  );
15 System.out.println(y.pop()  );
16 x.add(y.peek() +   y.peek());
17 System.out.println(x.remove()  );
18 System.out.println(x.peek()  );
19 System.out.println(y.pop() );
```

(5) **Stacks and Queues**

Write a method collapse that takes a Stack of integers as a parameter and that collapses it by replacing each successive pair of integers with the sum of the pair. For example, suppose a stack stores this sequence of values:

bottom (7, 2, 8, 9, 4, 13, 7, 1, 9, 10) top

Assume that stack values appear from bottom to top. In other words, 7 is on the bottom, with 2 on top of it, with 8 on top of it, and so on, with 10 at the top of the stack.

The first pair should be collapsed into 9 (7 + 2), the second pair should be collapsed into 17 (8 + 9), the third pair should be collapsed into 17 (4 + 13) and so on to yield:

bottom (9, 17, 17, 8, 19) top

As before, stack values appear from bottom to top (with 9 on the bottom of the stack, 17 on top of it, etc). If the stack stores an odd number of elements, the final element is not collapsed. For example, the sequence:

bottom (1, 2, 3, 4, 5) top

would collapse into:

bottom (3, 7, 5) top

with the 5 at the top of the stack unchanged.

You are to use one queue as auxiliary storage to solve this problem. You may not use any other auxiliary data structures to solve this problem, although you can have as many simple variables as you like. You also may not solve the problem recursively.

In writing your method, assume that you are using the Stack and Queue interfaces and the Stack and LinkedList implementations discussed in lecture. As a result, values will be stored as Integer objects, not simple ints.

Your method should take a single parameter: the stack to collapse.

(6) **Binary Trees**

Add the following items to a binary tree in the order given.

First : G J E Y R A X Z U H E

(7) **Binary Trees**

Write your tree from #1 above in order using each traversal type

- Prefix
- Postfix
- Infix

(8) **Binary Tree**

Write a method countLeftNodes that returns the number of left children in a tree. A left child is a node that appears as the root of the left-hand subtree of another node. For example, the following tree has four left children (the nodes storing the values 5, 1, 4, and 7):