Problem Set 3

This problem set is due **Tuesday, October 16** at **11:59PM**.

Code for problem 4, as well as a solution template for problem 4 and a LaTeXtemplate for the written problems, have been posted to the course website. Solutions should be turned in at `https://alg.csail.mit.edu` (our submission site).

Programming questions will be graded on a collection of test cases (including example test cases to help you debug). Unless you see an error message, *you will be able to see your grade immediately*. Your grade will be based on the number of test cases for which your algorithm outputs a correct answer within certain time and space bounds. You may submit as many times as you'd like, and only the final submission counts! **Therefore, make sure your final submission is what you want it to be.**

For written questions, full credit will be given only to correct solutions that are described clearly *and concisely*.

Problem 3-1. [20 points] **Trees and chains**

(a) [10 points] You are given a hash function $h(x) = x \,\%\, 10$ (where $\%$ denotes modulo operation). Collisions in the table are resolved by chaining. Sort the following sequences of length n in decreasing order of "badness", where badness is defined as length of the longest chain in the hash table. Partition the group into equivalence classes such that two sequences are in the same class if and only if their longest chain is equal. Assume that n is a large number and that the sequences are of format $x = (x_1, x_2, \ldots x_n)$.

Example Question:

$$x_i = 3 \tag{1}$$
$$x_i = i \tag{2}$$
$$x_i = i + 2 \tag{3}$$
$$x_i = 4 \tag{4}$$
$$x_i = 2i \tag{5}$$

Solution Format: Your answer to this problem should be a list of lists of integers. Each sublist should contain the indices of a set of functions that all have the same longest chain length. The order of the indices within the sublist does not matter. The sublists should be ordered from the longest/worst longest chain to the shortest/best.

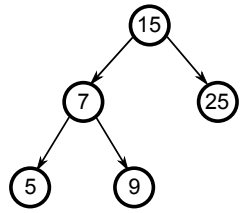
Include your answer in the file you upload, `pset3.py`. Here is an example of what you might include for full credit, for the question above:

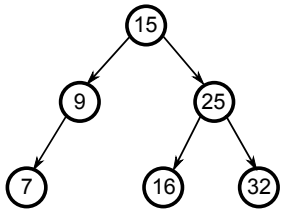
```
answer_problem_1_example = [[1, 4], [5], [2, 3]]
```

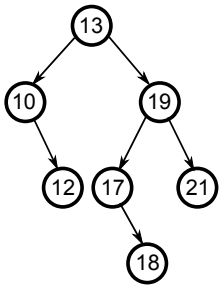
You do not need to show your work for this problem. Partial credit will be given.

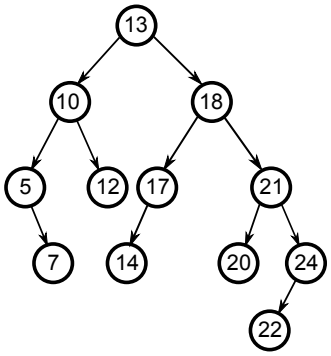
$$x_i = i \tag{1}$$
$$x_i = i \,\%\, 5 \tag{2}$$
$$x_i = i \,\%\, 6 \tag{3}$$
$$x_i = i^2 \tag{4}$$
$$x_i = i \,\%\, 10 \tag{5}$$
$$x_i = 15 * i \tag{6}$$
$$x_i = (15 * i) \,\%\, 7 \tag{7}$$
$$x_i = i \,\%\, 15 \tag{8}$$
$$x_i = 3 * i + 5 \tag{9}$$
$$x_i = (i \,\%\, 5 + 4) \tag{10}$$

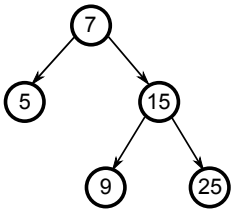
(b) [10 points] Consider, the following trees:

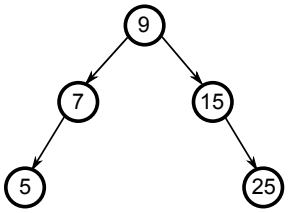
1)


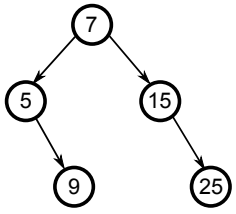
2)


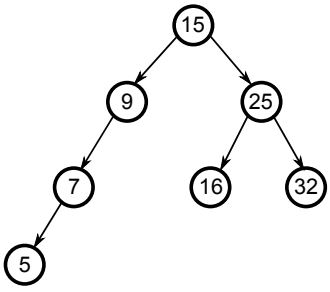
3)


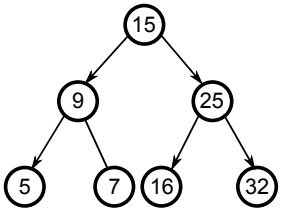
4)


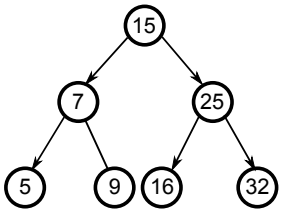
5)


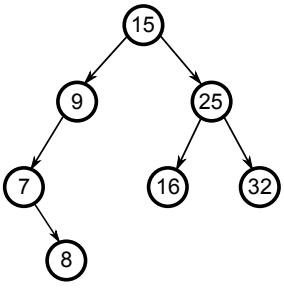
6)


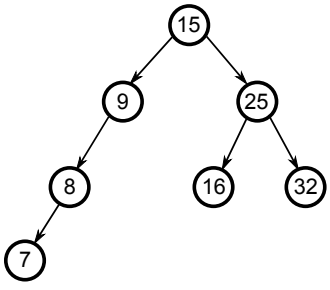
7)


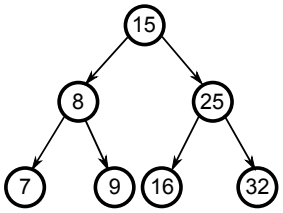
8)


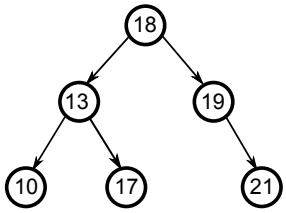
9)


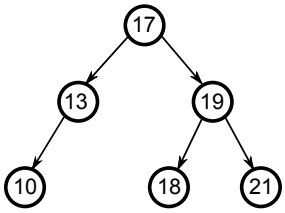
10)


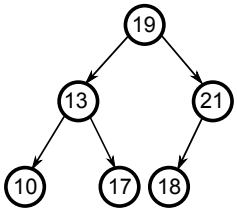
11)


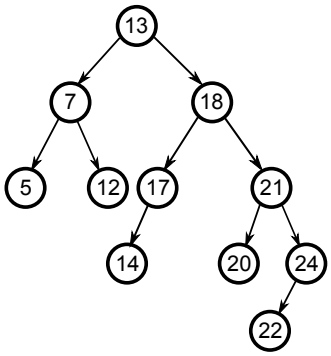
12)


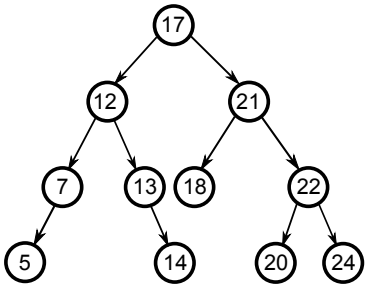
13) 

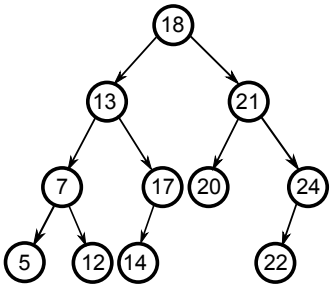
14) 

15) 

16) 

17) 

18) 

19) 

Now, answer the following questions:

1) Which trees are proper Binary Search Trees?

2) Which trees are proper Binary Search Trees that also satisfy the AVL property?

3) What tree is the result of RIGHT-ROTATE(15) on Tree 1?

4) What tree is the result of doing a BST-INSERT(5) on Tree 2?

5) What tree is the result of doing a BST-INSERT(5) on Tree 2, along with any necessary AVL balancing?

6) What tree is the result of doing a BST-INSERT(8) on Tree 2, along with any necessary AVL balancing?

7) What tree is the result of doing a BST-DELETE(12) on Tree 3, along with any necessary AVL balancing?

8) What is the result of doing a BST-DELETE(10) on Tree 4, along with any necessary AVL balancing?

Solution Format:

Your answer to parts 3-8 should be an integer between 1 and 19, inclusive, indicating which tree is the correct answer.

Your answer to parts 1 and 2 should be a set of integers between 1 and 19, inclusive.

Problem 3-2. [20 points] **Infiltrating the city of Dravrah**

A secret service led by Tim Beaver finally recognized your ninja skills and has hired you as their spy. Your first task is to find a way to infiltrate the city of Dravrah and establish radio communication with The Dome. The problem is that Dravrah has many secret services, each of which is monitoring a range of integer frequencies. To be precise, there are N secret services, each monitoring a subinterval $[a_i, b_i]$, where $1 \leq a_i < b_i \leq N^3$ are integers (As it turns out, 1 to N^3 is the range of all radio transmitters and receivers). Your predecessor, Jack Florey, managed to get the list of frequency ranges that are monitored. Some frequencies are monitored by more than one service. You would like to find a frequency that is unmonitored, or conclude that none exists. Tim Beaver is impatient, and N is big, so you need to act fast to keep your ninja street cred.

More formally, you are given N intervals, each defined by interval endpoints $[a_i, b_i]$, where $1 \leq a_i < b_i \leq N^3$. The intervals are given in arbitrary order and they may overlap. Your goal is to design an algorithm that will return an integer that is not contained in any of these intervals in $O(N)$ time, or conclude that the intervals cover the entire range.

You must also argue your algorithm's correctness and running time.

Problem 3-3. [20 points] **Impatient Rag**

You are walking down Mass Ave and you bump into a man, Rag. Rag came across a list of n strings and being a quirky guy, he thought it would be really cool to group the strings in a way that members of each group are anagrams of each other. However, the list is very long and he doesn't have the patience to do it by hand. Since being on MIT campus gives you the magic ability to solve the world's problems, Rag assumes you can solve his, too. Don't let him down!

The input you are given is a list of n strings. Each string has at most d characters, and the alphabet (list of possible characters) is finite. The expected output is a list of lists where members of each nested list are anagrams of each other. Design an algorithm that has an expected running time $O(n \cdot d)$, and argue its correctness and running time.

Example:
Input: `['aab', 'foo', 'aba', 'baa', 'oof', 'bar', 'baz']`
Output: `[['aab', 'aba', 'baa'], ['foo', 'oof'], ['bar'], ['baz']]`

Problem 3-4. [40 points] **Credit card number typos**

After having an awesome summer time at AMDtel, you decided it would be great to try out working for an online website which is collecting donations for the upcoming election. Unfortunately, political scientists often don't make good computer scientists, and the credit card numbers the website receives are often invalid. Looking into the issue, you find that for some reason, the error is often precisely that there is a pair of neighboring digits which are swapped (e.g. if the original card number is 12345678, you may receive 21345678 or 12345768, but not 21345768). Luckily, we have access to a database of potential correct credit card numbers, which we can match against our invalid ones. Because of rampant population growth, credit cards are now many digits long, although digits are still 0-9.

You need to write a function, `recover_credit_cards`, which takes two lists. The first list contains n valid, k-digit credit card numbers $v_0, v_1, \ldots v_{n-1}$. The second list contains m received k-digit numbers ($r_0, r_1, \ldots r_{m-1}$) that represent the numbers with an error in them. Your goal is to return a list s of length m where s_i is an index j of the valid credit card number v_j that r_i is derived from. Switching two adjacent digits that are equal is of course not an error, since no change is made, so you are guaranteed in this problem that r_i and v_{s_i} will never be identical.

Your code should have $O(k \cdot (n + m))$ expected running time for full credit. Thus your algorithm should scale well with all of the parameters. However, partial credit will be awarded for slower times.

Here are some tests which your function should pass:

```
recover_credit_cards(
    ['123', '678'],
    ['213', '132', '768']) == [0, 0, 1]

recover_credit_cards(
    ['3141593', '2718282', '1618034'],
    ['1681034', '2718228', '1341593']) == [2, 1, 0]
```