22C:16 Final Exam

May 11th, noon-2:00

This is an open notes exam. You have 2 hours to complete it.

1. **[70 points] On Dictionaries**

(a) **[30 points]** For one of your homework problems, you made a dictionary whose keys were legal 5-letter English words and the value corresponding to each key k was the list of legal 5-letter words that could be obtained from k by replacing exactly one letter. Here is a small portion of that dictionary. Suppose that this dictionary is called `D` and answer the following questions.

```
{"space": ["apace", "spice", "spade", "spake", "spare", "spate", "spacy"],
"xylem": [], "scold": ["scald"], "spacy": ["spicy", "space"],
"prize": ["price", "pride", "prime", "prise"]}
```

(i) What does the expression `D[D["space"][-1]][0]` evaluate to?

(ii) What is the output produced by the following code fragment?

```
for k, v in D.items():
    if len(v) <= 2:
        for w in v:
            print w
```

(b) [**40 points**] Your task for this problem is to write a function that "merges" a list of dictionaries. Here is an example to show you what I mean.

```
>>> D1 = {"hi": 10, "hello": 20}
>>> D2 = {"bye": 10, "hello": 30}
>>> mergeDict([D1, D2])
{'bye': [10], 'hi': [10], 'hello': [20, 30]}
```

In this example, `D1` and `D2` are two lists and `mergeDict` is a function that takes as argument a list `[D1, D2]` of dictionaries. The function `mergeDict` returns a new dictionary such that (i) every item that appears as a key in either `D1` or `D2` appears as a key in the new dictionary and (ii) if an item appears as a key in both `D1` and `D2`, then its corresponding value in the new dictionary is the list of the corresponding values in `D1` and `D2` (consider for e.g. the key "hello"). While in my example the argument to `mergeDict` was a list of length 2, `mergeDict` should work no matter how many dictionaries there are in the given list.

Notes: My solution to this problem has 9 lines of code. Use this as a rough estimate of how much code you want to write - if you seeing yourself writing too much, you should rethink your approach.

2. **[70 points] On Recursion**

(a) **[30 points]** Consider the following function that computes and returns the n^{th} Fibonacci number. Recall that the Fibonacci sequence is

$$1, 1, 2, 3, 5, 8, 13, 21, \ldots$$

in which term n, for $n > 2$, is obtained by adding terms $(n-1)$ and $(n-2)$.

```
def fibonacci(n):
    if n == 1 or n == 2:
        return 1

    x = fibonacci(n-1)
    print x
    y = fibonacci(n-2)
    print y
    return x + y
```

(i) What is the output produced (from the print statements) when we make the call

`fibonacci(6)`

Even though each number in the output appears in a different line, to save space you can write down the numbers in the same line, separated by white space.

(ii) When `fibonacci` is called with argument 1 or 2, no lines of output are produced. When `fibonacci` is called with argument 3, two lines of output are produced. Complete the following table by writing down the number of lines of output produced for `n` $= 7, 8, 9, 10$. I have supplied values for `n` $= 1, 2, \ldots, 6$.

Value of `n`	No. of lines of output
1	0
2	0
3	2
4	4
5	8
6	14
7	
8	
9	
10	

Hint: You don't want to solve this problem by tracing the function for each value `n` $= 7, 8, 9, 10$. Instead, you should notice a pattern (similar to the Fibonacci sequence pattern) in the number of lines of output and use this to complete the table.

(b) **[40 points]** Write a recursive function called `isPalindrome` that determines if a given string is a *palindrome*. The function should return `True` if the given string is a palindrome and `False` otherwise. Examples of palindromes are "dad", "abba", etc.

A palindrome is any string s that satisfies two conditions:

(i) the first and last characters of s are identical and

(ii) the string obtained by stripping off the last and first characters from s is also a palindrome.

This definition of a palindrome is essentially a recursive algorithm for determining if s is a palindrome. I want you to implement this algorithm for the function `isPalindrome`.

Notes. (i) You will get no points if you implement an algorithm different from the one described above and (ii) Don't forget the base cases.

3. **[60 points] On Classes and Objects.**

(a) **[30 points]** Answer the two questions below based on the following class implementation.

```
class goofy():

    def __init__(self, L):
            self.info = []
            for i in range(len(L)):
                try:
                    self.info.append(str(i)*L[i])
                except TypeError:
                    print "Use a list of non-negative integers."

    def display(self):
        for e in self.info:
            print e
```

(i) What output do you get from trying the following at a Python shell?

```
>>> x = goofy([7, 1, 2])
>>> x.display()
```

(ii) What output do you get from trying the following at a Python shell?

```
>>> x = goofy(["hello", 4, 0, -10, 3])
>>> x.display()
```

(b) **[30 points]** Implement a class called `biasedCoin`. Each instance of the `biasedCoin` class represents a coin (e.g., a penny) that can be tossed. However, we are talking about "biased" coins here and therefore the two outcomes of a coin toss are *not* equally likely and associated with each instance of the `biasedCoin` class is a "bias," which is a real number between 0 and 1 representing the probability that the outcome will be "heads" when the coin is tossed. Here is an example of how I would construct an instance of the `biasedCoin` class:

```
c = biasedCoin(0.2)
```

This assignment creates an instance of the `biasedCoin` class called `c`, with "bias" 0.2, i.e., when the coin `c` is tossed it with return "heads" with probability 0.2 and "tails" with probability 0.8.

The `biasedCoin` class should provide two methods:

(i) A constructor for the `biasedCoin` class that takes a real number as an argument. The constructor should raise a `ValueError` if the argument provided to the constructor is not a real number or it is a real number, but it is smaller than 0 or larger than 1.

(ii) A method called `toss` that returns a string `"H"` or a string `"T"` depending on whether the outcome of the coin toss was heads or tails. The probability that `"H"` is returned should be equal to the "bias" of the instance and `"T"` should be returned with probability equal to one minus this "bias." I would call the `toss` function as follows.

```
print c.toss()
```

Suggestions. (i) My suggestion would be to use a single `float` attribute to keep track of the "bias" of the coin. (ii) I would also suggest the use of the `random.random()` that returns a real number in the range $[0, 1)$.