Online Practice Final Answers

TRUE / FALSE

For \( f(n) = 2000* (\log n)^{2} + 2*n^{5} \), the order of \( f(n) \) is \( O(n^{5}) \).
True

You can create a dictionary where the key is a list type. For example, list \( a = [4, 3] \) and dictionary \( d = {} \). Once we assign \( d[a] = 3 \), we will be able to look up list \( a \) in the dictionary at any later time to get the value 3.
False

A Python class is an example of data abstraction.
True

A thrown exception that is not handled in the code always leads to program termination.
True

In Python, classes cannot be used as a parameter of a function.
False

The order of growth of the running time of a program is most apparent when testing with small inputs.
False

All \( O(1) \) functions take exactly the same amount of time to run.
False

\( 2N+99999999 \) and \( 99999999N+2 \) are both \( O(N) \).
True

In Python, you can instantiate many objects of the same class.
True

A Python subclass inherits the variables but not the methods of the parent class.
False

CODING

Assume \( s \) is a string of lower case characters. Write a function called \( \text{count\_bobs} \) that returns the number of times the string 'bob' occurs in \( s \). For example:

if \( s = 'azcbobobegghakl' \), then your program should return 2
if \( s = 'bo' \), then your program should return 0
if \( s = 'abbobbbob' \), then your program should return 2
if \( s = 'bobobobobobo' \), then your program should return 4

Answer:

```python
def count_bobs(s):
    numBobs = 0
    for i in range(1, len(s)-1):
        if s[i-1:i+2] == 'bob':
            numBobs += 1
    return numBobs
```
DEBUGGING: Answer:

class Frob(object):
    def __init__(self, name):
        self.name = name
        self.before = None
        self.after = None

    def setBefore(self, before):
        # example: a.setBefore(b) sets b before a
        self.before = before

    def setAfter(self, after):
        # example: a.setAfter(b) sets b after a
        self.after = after

    def getBefore(self):
        return self.before

    def getAfter(self):
        return self.after

    def getName(self):
        return self.name

    def insert(self, newFrob):
        if self.getName() < newFrob.getName():
            lastFrob = self
            currentFrob = self.getAfter()
            while currentFrob != None and currentFrob.getName() < newFrob.getName():
                nextFrob = currentFrob.getAfter()
                lastFrob = currentFrob
                currentFrob = nextFrob
            if currentFrob != None:
                lastFrob.setAfter(newFrob)
                newFrob.setBefore(lastFrob)
                newFrob.setAfter(currentFrob)
                currentFrob.setBefore(newFrob)
            else:
                lastFrob.setAfter(newFrob)
                newFrob.setBefore(lastFrob)
        else:
            lastFrob = self
            currentFrob = self.getBefore()
            while currentFrob != None and currentFrob.getName() > newFrob.getName():
                nextFrob = currentFrob.getBefore()
                lastFrob = currentFrob
                currentFrob = nextFrob
            if currentFrob != None:
                lastFrob.setBefore(newFrob)
                newFrob.setAfter(lastFrob)
                newFrob.setBefore(currentFrob)
                currentFrob.setAfter(newFrob)
            else:
                lastFrob.setBefore(newFrob)
                newFrob.setAfter(lastFrob)

    def findFront(self):
        front = self
        while front.getBefore() != None:
            front = front.getBefore()
        return front