Name: ______________________________ Recitation: ______ Andrew Id: ________________

# 15-112 Spring 2018 Quiz 3

Up to 20 minutes. No calculators, notes, books, or computers. Do not use lists or recursion. Show your work!

1. (30 points) **Short Answer:** Identify three of the rules in the 15-112 style guide and write a short piece of code that follows all three rules. The code should loop over the characters in a string and count the number of question marks that appear in the string.

2. (20 points) **Short Answer:** Below we have two implementations of isPrime. One is correct, and one has a bug. Write a single test case that will pass when run with the correct program and throw an error when run with the incorrect one. Comment the test with a sentence explaining why the test should fail on one of the programs.

```
def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, int(round(n**0.5))):
        if n % i == 0:
            return False
    return True
```

```
def isPrime(n):
    if n < 2:
        return False
    for i in range(2, n-1):
        if not (n % i > 0):
            return False
    return True
```

3. (20 points) **Debugging**: The piece of code shown below is supposed to implement the function isPalindrome (which we went over in class), but it has a bug. The bug occurs when isPalindrome is run on "abba"; the function should return True with that argument, but instead it throws the error shown below. Circle the part of the code that needs to be changed and write in the box below the code what you would change the code to.

```
1: def isPalindrome(s):
2:     for i in range(len(s)):
3:         if (s[i] != s[len(s)-i]):
4:             return False
5:     return True
```

```
Traceback (most recent call last):
  File "quiz3.py", line 7, in <module>
    print(isPalindrome("abba"))
  File "quiz3.py", line 3, in isPalindrome
    if (s[i] != s[len(s)-i]):
IndexError: string index out of range
```

4. (30 points) **Free Response:** Two words are considered anagrams if they contain the same letters, but in potentially different orders. We'll then define **almost-anagrams** to be two words that either a) are anagrams, or b) are anagrams apart from one extra letter that occurs in one of the two words. For example, "melon" and "lemon" are anagrams (and therefore almost-anagrams as well). Additionally, "class" and "slacks" are almost-anagrams, since "slacks" has all the letters of "class", plus a k. On the other hand, "dog" and "bog" are not almost-anagrams, since they are two letters apart.

   Write the function areAlmostAnagrams(s1, s2) which takes two strings as arguments and returns True if they are almost-anagrams and False otherwise. You are welcome to write helper functions if needed. You may **not** assume that areAnagrams has already been written.