

fullName:_____ andrewID:_____ section:_____

15-112 S25
Quiz3 version B

Read these instructions carefully before starting:

1. Quiz versions are color-coded. You must have a different version (color) of this quiz than the students sitting to your left and right.
2. Stop writing and submit the entire quiz when instructed by the proctor.
 - Do not unstaple any pages.
 - You must submit the entire quiz with all pages intact.
3. Do not discuss the quiz with anyone else until after 5pm.
 - This applies to everyone, including students in either lecture.
4. Do not use your own scrap paper.
 - You should not need scrap paper, there is plenty of room for you on the quiz.
 - However, if you absolutely must use scrap paper, raise your hand and we will provide some. Then, you must write your andrew id clearly on the scrap paper, and hand in the scrap paper with your paper quiz. We will not grade anything on your scrap paper.
5. You may not ask questions during the quiz.
 - The one exception is for English-language clarifications.
 - If you are unsure how to interpret a problem, just take your best guess.
6. Do not use any concepts (including built-in functions) not covered in the notes through week 3 or beyond unit 2.
 - Do not use lists, tuples, dictionaries, sets, or recursion.
7. Do not hardcode your solutions.
 - We may test your code using additional test cases.
 - Hardcoding will receive zero points.
8. Assume `almostEqual(x, y)` and `rounded(n)` are both supplied for you.
 - You must write all other helper functions you wish to use, unless we specify otherwise.
9. Good luck!

Code Tracing (CT) [30 pts, 10 pts each]

For each CT, indicate what the code prints

Place your answer (and nothing else) in the box below the code.

Note: If floats occur in these CTs, they will have no more than one digit after the decimal point.

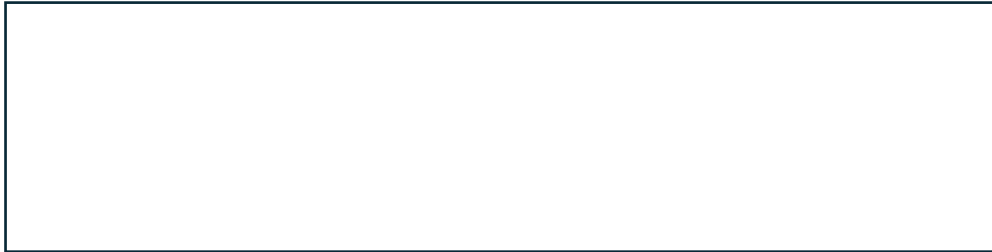
CT1:

```
import string
def ct1(s):
    return s[:1] + '-' + s[-2:] + '-' + s[1:6:2]

print(ct1(string.ascii_uppercase))
```


CT2:

```
def ct2(s):  
    r = ''  
    s = s.replace('a,', 'f') # Don't miss the comma  
    for t in s.split(','):   
        if t.isalpha():  
            r += t  
        else:  
            r *= int(t)  
    return r[::-1]  
  
print(ct2('a,b,2,e'))
```



CT3:

```
def ct3(s):  
    r = ''  
    for i in range(len(s)):  
        j = ord(s[i]) + i  
        if j > ord('Z'):  
            j = ord('A') + i  
        r += chr(j)  
    return f"{r}{s.find('Y')}{s.find('A')}"  
  
print(ct3('WXYZ'))
```



Free Response / FR1: largestInt [30 pts]

Reminder: you cannot use lists to solve this problem.

You can use `s.split()` or `s.splitlines()`, but only if you **loop** over the result **and** do not store the result in a variable.

Write the function `largestInt(s)` that takes a possibly-empty string `s` that is guaranteed to contain words and/or non-negative integers each separated by one space, and returns the largest integer in the string, or `None` if the string contains no integers.

For example:

```
assert(largestInt('1 dog and 2 cats and 0 birds') == 2)
assert(largestInt('23 dogs 2 cats and 0 birds') == 23)
assert(largestInt('no dogs or cats or birds') == None)
assert(largestInt('0 dogs') == 0)
assert(largestInt('1 22 333 4444 333 22 1') == 4444)
assert(largestInt('') == None)
```

Begin your FR1 answer here or on the next page.

Begin or continue your answer to FR1 here:

Free Response / FR2: gradeQuiz [40 pts]

Reminder: you cannot use lists to solve this problem.

You can use `s.split()` or `s.splitlines()`, but only if you loop over the result and do not store the result in a variable.

Background: for this problem, you will grade multiple choice quizzes. You will be given a spec that is a multiline string like this:

```
spec = '''KEY: AB
Ann: AB
Bob: BB
Carl: AA'''
```

The first line of the spec is the answer key. It always starts with 'KEY: ' followed by the correct answers to each problem. There are two problems in the example above:

- A is the correct answer to the first problem.
- B is the correct answer to the second problem.

You can assume every answer is a single uppercase letter.

Each line after that starts with a student's name, followed by a colon, then a space, then that student's responses. You can assume every student answered every question with a single uppercase letter. Given this spec, you need to produce a grade report like this:

```
result = '''GRADEBOOK
Ann: 2/2
Bob: 1/2
Carl: 1/2
'''
```

Each line starts with the student's name (in the same order as the spec), followed by a colon and a space, then the number they got correct, then a slash (' / '), then the total number of questions.

With that, write the function `gradeQuiz(spec)` that takes a spec as just described, and returns a grade report again as just described.

Hint: When you read the first line, we recommend storing the key as its own string, since you will need it frequently. (Plus, one way to know if you are looking at the key line or a student line is to check if you have assigned something meaningful to the key string yet.)

Hint: We recommend writing AT LEAST one helper function, perhaps more. For example, you might write `countMatches(A, B)`, which takes two same-length strings and returns how many characters match at each index. For example, `countMatches('ABAB', 'BBAC')` would return 2 because the middle two characters are the same in both strings.

Here are some test cases for you:

```
        spec = '''KEY: AB
Ann: AB
Bob: BB
Carl: AA'''

        result = '''GRADEBOOK
Ann: 2/2
Bob: 1/2
Carl: 1/2
'''
        assert(gradeQuiz(spec) == result)

        spec = '''KEY: ACAB
Ann: ACAB
Bob: EEEE
Carl: ACAA
Daniel: ABBE'''

        result = '''GRADEBOOK
Ann: 4/4
Bob: 0/4
Carl: 3/4
Daniel: 1/4
'''
        assert(gradeQuiz(spec) == result)
```

Begin your FR2 answer on the next page.

Begin your answer to FR2 here:

Continue your answer to FR2 here:

Bonus Code Tracing (BonusCT) [Optional, 2 pts each]

Bonus problems are not required.

For these CTs, indicate what the code prints.

Place your answer (and nothing else) in the box below the code.

BonusCT1:

```
def bonusCt1():
    uu = ('inches,feet,yards,'+
          'miles,hours,days,months,years')
    k = 0
    for c in 'aeiou': k = max(k, uu.count(c))
    def u(i):
        while i > 0:
            for u in (uu.split(',')):
                i -= 1
                if not i: return u
    d = False
    for p in range(1, k):
        n = 7 * 2**p
        s = u(p)[3-p] + u(p)
        t = f'{n}-{s}'
        if d: return t
        d = 'yy' in t
    print(bonusCt1())
```

BonusCT2:

```
def bonusCt2(n):
    # Hint: int(s, b) converts s (a number written in
    #     base-b) to an int.
    # This uses try/except, which we have not yet learned,
    #     but you should be able to figure out what it does!
    #     (It has something to do with runtime errors...)
    t = 0
    while n:
        try:
            t += 1
            z = int(str(t), 2)
            n -= 1
        except:
            pass
    return t
print(bonusCt2(5))
```

