fullName: _____ andrewID: _____ section: ____

15-112 S26 Quiz4 Version B
Time: 30 Minutes

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use material we have not yet covered. Thus, do not use lists, tuples, sets, dictionaries, OOP, or recursion.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than the specified time noted above on this assessment.

Style [Yes/No, 5 pts]

Which of the following are in our style checklist in the Style notes? Choose ALL that apply. Indicate your answer by filling in the dot.

A) Try to use magic numbers.

◯ Yes

◯ No

B) Use meaningful variable names.

◯ Yes

◯ No

C) Use comments but not too heavily.

◯ Yes

◯ No

D) Use well-chosen and well-named helper functions.

◯ Yes

◯ No

E) Use global variables sparingly or not at all.

◯ Yes

◯ No

F) Avoid using floats.

○ Yes

○ No

Fill in the Blank [5 pts]

Fill in the blank in g(n) so that f(n) == g(n) for all integer values of n:

```
def f(n):
    if n < 0:
        n = -n
    r = 0
    while n > 0:
        if n % 4 == 0:
            r += n
        n -= 1
    return r

def g(n):
    r = 0
    for x in range(_____________________________________________________):
        r += x
    return r
```

Code Tracing [10 pts, 5 pts each]

For each of the following, indicate what the code prints. Place your answer (and nothing else) in the box below each block of code. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

CT1: [5 pts]

```
def ct1(n):
    s = ''
    i = 0
    while n > 0:
        d = n%10
        n //= 10
        i = (i + d) % 10
        e = chr(ord('0') + i)
        s = e + s
    return repr(s)

print(ct1(985))
```

CT2: [5 pts]

```
def ct2(s, t):
    n = 0
    for c in t:
        n = 10*n + s.find(c)
    return n

print(ct2('BCDB', 'CDBA'))
```

FR1: nearestPowerOfTen(n) [20 pts]

Note:

- For full credit, do not use loops or strings here.
- For half credit, you can use loops or strings if you wish.

The powers of 10 are: 1, 10, 100, 1000, and so on.

With that, write the function nearestPowerOfTen(n) that takes an arbitrary Python value n, and if n is a float or int, it returns the nearest power of 10 to n, with ties going to the smaller value. If n is not a float or int, it returns None.

For example:

```
assert(nearestPowerOfTen(5) == 1)
assert(nearestPowerOfTen(5.5) == 1) # 4.5 away from both 1 and 10
assert(nearestPowerOfTen(6) == 10)
assert(nearestPowerOfTen(99) == 100)
assert(nearestPowerOfTen(1234) == 1000)
assert(nearestPowerOfTen(-6) == 1)
assert(nearestPowerOfTen('Do not crash here') == None)
```

Write your solution on the next page:

FR2: nthPrimish(n) [30 pts]

Note:

- You may use loops but do not use strings here.
- You may use isPrime here without writing it.

Background: we will say that an integer n is "primish" (a coined term) if:

- n is not prime, and
- n is odd, and
- n is the average of two primes.

For example, 25 is primish because it is the average of the primes 3 and 47. For what it's worth, 25 is also the average of other prime pairs, like 19 and 31.

The first few primish numbers are: 9, 15, 21, 25, 27, ...

With that, write the function nthPrimish(n) that takes a non-negative int n and returns the nth primish number, where nthPrimish(0) returns 9.

For example:

```
assert(nthPrimish(0) == 9)
assert(nthPrimish(1) == 15)
assert(nthPrimish(2) == 21)
assert(nthPrimish(3) == 25)
```

Write your solution on the next page:

FR3: leastCommonVowels(s) [30 pts]

Note:

- This string contains all the vowels: 'AEIOU'.

With that, write the function leastCommonVowels(s) that takes a possibly-empty string s and returns a string containing the vowels that occur the least often in s (in a comma-separated uppercase string in alphabetical order) or None if there are no vowels in s. Your function should be case-insensitive, so treat 'a' and 'A' the same.

For example:

```
assert(leastCommonVowels('Airing') == 'A')       # 1 A and 2 I's
assert(leastCommonVowels('Amazing') == 'I')      # 2 A's and 1 I
assert(leastCommonVowels('a BIG fair') == 'A,I') # 2 A's and 2 I's
assert(leastCommonVowels('lynx ** zzz') == None) # No vowels
```

Write your solution on the next page:

BonusCT: These CTs are optional, and intended to be very challenging. They are worth very few points. Indicate what the following code prints. Place your answers (and nothing else) in the boxes to the right of each code block. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

bonusCT1: [1pt]

```
def bonusCt1(s):
    def f(s):
        for i in range(len(s)):
            for j in range(i+1, len(s)):
                if s[i] < s[j]:
                    s = s[:i] + s[j] + s[i+1:j] + s[i] + s[j+1:]
        return s
    def g(s):
        t = ''
        for c in s: t += chr(ord(c)+1)
        t = t[1:]
        return t[::-1]
    t = ''
    while s:
        t += s[0]
        s = f(g(s))
    return t
print(bonusCt1('EAR'))
```

bonusCT2: [1pt]

```
def bonusCt2(d):
    e = 0
    while not chr(e).isdigit() or not int(chr(e)): e += 1
    # hint: 'd' > 'Q' > '7'
    for k in range(ord('Z'), 0, -1):
        c = chr(k)
        if c.isdigit() and (k - e) % d == 1:
            d = 10*d + int(c)
    return d
print(bonusCt2(4))
```