fullName:__________________ andrewID:__________________ section:___

15-112 F24
Quiz2 version B (20 min)

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 4pm.
    - 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 2.
    - Do not use strings, 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) [20 pts, 10 pts each]
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

CT1:

```
def ct1(x):
    while True:
        print(x)
        if x % 2 == 0:
            x += 1
            continue

        if x % 2 == 1:
            x *= 2
            if x > 9:
                break

ct1(3)
```

CT2:

```
def ct2(n):
    for x in range(n, 0, -2):
        y = 1
        while y < x:
            print(f'{x}:{y}')
            y += 2

print(ct2(4))
```

Free Response / FR1: isPrime [15 pts]

Write the function isPrime(n) that takes a possibly-negative integer n and returns True if n is prime, and False otherwise.

Note: We will not grade isPrime on its efficiency. Any correct implementation will receive full credit.

Here are some test cases:

```
    assert(isPrime(1) == False)
    assert(isPrime(2) == True)
    assert(isPrime(7) == True)
    assert(isPrime(8) == False)
    assert(isPrime(9) == False)
    assert(isPrime(0) == False)
    assert(isPrime(-1) == False)
    assert(isPrime(-7) == False)
```

Begin your answer to FR1 here:

Free Response / FR2: isSwappy [50 pts]

We will say that an integer n is “swappy” (a coined term) if:

- n is positive, and
- n is not prime, and
- n contains exactly 2 odd digits, and
- If we swap the two odd digits in n we get a prime number

For example, consider n == 749:

- 749 is positive.
- 749 is not prime (it equals 7 * 107).
- 749 contains exactly 2 odd digits (7 and 9).
- If we swap the 7 and 9, we get 947, which IS prime.

Thus, 749 is swappy.

The first 10 swappy numbers are:

- 35, 91, 95, 125, 145, 215, 217, 275, 301, 305

With that, write the function isSwappy(n) that takes a possibly-negative int n and returns True if it is swappy and False otherwise.

Note: for this problem, assume that these functions are already written for you, so you can use them without writing them yourself:

- `digitCount(n)`
- `getKthDigit(n, k)`
- `setKthDigit(n, k, d)`
- `isPrime(n)`

Here are some test cases:

```
    assert(isSwappy(35) == True)
    assert(isSwappy(91) == True)
    assert(isSwappy(71) == False)  # 71 is prime
    assert(isSwappy(-5) == False)
    assert(isSwappy(93) == False)  # 39 = 13 * 3
    assert(isSwappy(110) == False)
    assert(isSwappy(119) == False) # three odd digits
```

Begin your answer to FR2 here:

Free Response / FR3: nthSwappy [15 pts]

Write the function nthSwappy(n) that takes a non-negative integer n, and returns the nth swappy number, so that nthSwappy(0) return 35.

Here are the first 10 swappy numbers:

- 35, 91, 95, 125, 145, 215, 217, 275, 301, 305

Note: for this problem, assume that these functions are already written for you, so you can use them without writing them yourself:

- `isSwappy(n)`

Begin your answer to FR3 here:

Bonus Code Tracing (BonusCT) [Optional, 2 pts]
Bonus problems are not required.
For each CT, indicate what the code prints
Place your answer (and nothing else) in the box below the code.

BonusCT1:

```
def  bonus(x):
    a, b, c = x%10, x//10%10, x//100%10
    x = 100*(b-1)+10*(c-1)+(a-1)
    return x if (min(a,b,c)==0) else bonus(x)
print(bonus(635))
```