fullName:_____andrewID:_____ recitationLetter:_____

15-112 S23

Quiz2 version B (25 min)

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in week 2 / the loops section of unit 2.
- You may not use strings, lists, indexing, tuples, dictionaries, sets, or recursion.
- We may test your code using additional test cases. Do not hardcode.
- 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.

CT1: Code Tracing [10pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(n):
    z = n
    for x in range(1, n, 4):
        z = 10*z + x
        for y in range(n, n + 2):
            z = 10*z + y
    return z

print(ct1(8))
```

CT2: Code Tracing [12pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct2(xMax, yMax):
    for x in range(1, xMax):
        y = yMax
        while x <= y:
            if x == y:
                print(y+x)
            elif x == 2 and y == 3:
                break
            else:
                print(y+x, end=" ")
            y -= 1
    return x+y
ct2(4, 5)
```

Free Response 1: isGregorNumber(n) [40pts]

We will say that a number is a "Gregor number" (a made-up term) if it is a positive integer where the number of times each digit appears is less than or equal to the digit itself. For example, the digit 1 must not appear more than once, the digit 3 must not appear more than 3 times, and the digit 6 cannot appear more than 6 times. 0 cannot appear in any Gregor number. Here are some example Gregor numbers: 1, 333122, 492236134, 23213

However, these are not Gregor numbers because at least one digit appears too many times: 11, 2232, 331233, 10, 555555

These are not Gregor numbers either because they are not positive: 0, -1, -42

With this in mind, write the function isGregorNumber(n) that takes an integer n and returns True if n is a Gregor number, and False otherwise.

Do not use strings, lists, or other prohibited concepts or functions which are not in the week 1 or week 2 notes.

Begin your FR1 answer here or on the following page

You may begin or continue your FR1 answer here

Free Response 2: nthGregorPrime(n) [38 points]

A "Gregor prime" is a number that is **both a Gregor number and is prime.** Using the definition of "Gregor numbers" from FR1, write the function nthGregorPrime(n) that takes a non-negative integer and returns the nth Gregor prime. For this problem, assume isGregorNumber(n) is written properly, given the definition in FR1 (i.e. you do not need to rewrite isGregorNumber(n) for this question, and you may simply call it and assume it is correct.)

The 0th Gregor prime is 2. Here are some additional test cases:

```
assert(nthGregorPrime(0) == 2)  # Watch out for off-by-one errors
assert(nthGregorPrime(1) == 3)
assert(nthGregorPrime(4) == 13)
assert(nthGregorPrime(5) == 17)
assert(nthGregorPrime(30) == 167)
assert(nthGregorPrime(500) == 5237)
```

We strongly recommend you write an isPrime(n) helper function. Do not use strings, lists, or other prohibited concepts or functions which are not in the week 1 or week 2 notes.

Begin your FR2 answer here or on the following page

You may begin or continue your FR2 answer here, if you wish

bonusCT: Code Tracing [2pts]

This question is optional, and intentionally quite difficult. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def f(x, d):
    z = 0
    while x:
        z, x = z or not (x%10-d), x//10
    return z

def bonusCt1(r):
    for i in range(11):
        d = (i**2)%10
    if not f(r, d):
        r = 10*r + d
    return r
print(bonusCt1(4))
```