Name: _____ Andrew Id: _____

15-112 Spring 2025 Quiz 3

Up to 25 minutes. No calculators, no notes, no books, no computers. Show your work!
Do not use strings, lists, dictionaries, try/except, or recursion on this quiz.

1. **Code Tracing**: Indicate what the following program prints. Place your answer (and nothing else) in the box next to the code.

   (a) (5 points) CT1

```
def h1(x):
    if (x % 2 == 0):
        return x * 2
    else:
        return x // 2

def ct1(n):
    total = 0
    while (total <= 42):
        total += h1(n)
        print("total:",total)
        if (total % 10 == 4):
            n = n - 4
            continue
        print("n is", n)
        n -= 1
    return total

print(ct1(12))
```

   (b) (5 points) CT2

```
def ct2(x):
    count = 0
    target = x
    for i in range(7,1,-1):
        if count == target:
            print("meet", count)
            break
        m = i
        while m < 20:
            count += i
            m = m * 2
        print("miss", count)
        count += 2
    return count

count = 1
ct2(42)
print(count)
```

2. (10 points) **Free Response: Next EqualizedFortyTwo Number**

An **EqualizedFortyTwo** number (coined term) is an integer that satisfies all of the following:

- It is positive.
- The sum of its even digits equals the sum of its odd digits.
- It contains the digit-sequence 42 in its decimal representation.

For example, 25423 is an *EqualizedFortyTwo* number because it is positive, contains 42 when read left to right (the third and fourth digits), and the sum of its even digits ($2 + 4 + 2 = 8$) equals the sum of its odd digits ($5 + 3 = 8$). On the other hand, 2542 is not an *EqualizedFortyTwo* number because, although it contains 42, the sum of its even digits is 8, while the sum of its odd digits is 5. The number 341211 is **not** an *EqualizedFortyTwo* because, although the sum of its even digits (4+2=6) is equal to the sum of its odd digits (3+1+1+1=6), it doesn't contain a 42 digit-sequence in it. For your reference, the first five *EqualizedFortyTwo* numbers are: 1425, 1542, 3342, 3423, 4215.

With that in mind, write the function `nextEqualized42(n)`, which takes a positive integer `n` and returns the smallest *EqualizedFortyTwo* number that is greater than or equal to n.

Below are a few examples:

- `nextEqualized42(1)` returns **1425**, the smallest *EqualizedFortyTwo* number ≥ 1.
- `nextEqualized42(100)` also returns **1425**, the smallest *EqualizedFortyTwo* number ≥ 100 (since no *EqualizedFortyTwo* numbers exist between 100 and 1424).
- `nextEqualized42(1542)` returns **1542** (since 1542 is already *EqualizedFortyTwo*).
- `nextEqualized42(2000)` returns **3342**.
- `nextEqualized42(3450)` returns **4215**.

Hints:

- Define a separate boolean helper function `isEqualized42(n)` to check if a number is *EqualizedFortyTwo*. Then, in your `nextEqualized42` function, call this helper each time you need to check the condition.
- Even if you cannot fully implement `isEqualized42`, you can write `nextEqualized42` as if `isEqualized42` already works. This will still demonstrate the structure of your solution and may earn you partial credit.

Note: You may not use strings in this problem!! A solution that uses strings will receive 0 points.

Additional Space for Answer to Question 2