Name: _____ Andrew Id: _____

15-112 Fall 2025 Quiz 2

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

```
def f(x, m):
    for y in range(m, m+4, 2):
        x += y
    return x

def ct1(m):
    x = 8
    while x//3 > 1:
        print("x = ", x)
        x -= 2
        print("f = ", f(x,m))
    return x

print(ct1(4)) # prints 5 lines
```

2. (6 points) **Free Response**: Collatz Sequence

   Write the function `collatzCount(n)` that takes a positive integer `n` and returns the number of steps required for `n` to reach `1` by repeatedly applying the following rules [1] :

   - If `n` is even, replace it with `n // 2`.
   - If `n` is odd (and greater than 1), replace it with `3*n + 1`.

   The count should include all steps until `n` becomes exactly `1`. You may assume that `n >= 1`.

   For example:

   - `collatzCount(1)` returns `0`, because it is already 1.
   - `collatzCount(6)` returns `8`, because the sequence is `6 -> 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1`, which takes 8 steps.
   - `collatzCount(11)` returns `14`, because the sequence has 14 steps before reaching 1.

   More examples:

```
assert collatzCount(2) == 1  # 2 -> 1
assert collatzCount(3) == 7  # 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
assert collatzCount(4) == 2  # 4 -> 2 -> 1
assert collatzCount(5) == 5  # 5 -> 16 -> 8 -> 4 -> 2 -> 1
```

[1] This problem is based on the famous Collatz conjecture (also known as the 3n + 1 problem). It is an unsolved problem in mathematics: although no one has proven it, every number tested so far eventually reaches 1 when following these rules.

3. (8 points) **Free Response**: nthTwoFactorish

We will say that a positive integer is *two-factorish* (a made-up term) if:

- It has at least two digits, and
- the ratio between each pair of consecutive (neighboring) digits is exactly 2 or $\frac{1}{2}$ (in other words, each digit is either double or half the previous one).

For example:

- `12` is two-factorish since $2 = 2 \times 1$.
- `21` is two-factorish since $1 = \frac{1}{2} \times 2$.
- `24` is two-factorish since $4 = 2 \times 2$.
- `42` is two-factorish since $2 = \frac{1}{2} \times 4$.
- `36` is two-factorish since $6 = 2 \times 3$.
- `4248` is two-factorish since $2 = \frac{1}{2} \times 4$, $4 = 2 \times 2$, and $8 = 2 \times 4$.

Here are the first 15 two-factorish numbers: 12, 21, 24, 36, 42, 48, 63, 84, 121, 124, 212, 242, 248, 363, 421.

With that, write the function `nthTwoFactorish(n)` that takes a non-negative integer `n`, and returns the nth two-factorish number.

For example:

- `nthTwoFactorish(0)` returns `12`.
- `nthTwoFactorish(3)` returns `36`.
- `nthTwoFactorish(11)` returns `242`.

Suggestion: As in lecture, it may help to first write a helper function `isTwoFactorish(m)` that returns `True` if `m` is two-factorish and `False` otherwise. Then, in `nthTwoFactorish(n)`, search through the positive integers, counting how many are two-factorish, until you find the nth one.