Name: ______________________________ Andrew Id: ____________________

15-112 Spring 2026 Quiz 2

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

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

   (a) (5 points) CT1

```python
def CT1(n):
    for i in range(1, n, 5):
        for j in range(i//2, 1, -1):
            print(n)
            if n > 10:
                n %= 9
                print(f"{n} < 10")
            elif n == 2:
                print("n = 2")
    for i in range(n, 1):
        print("Hit")

CT1(11)
```

2. (8 points) **Free Response**: Nth Swappy Number

   Do not use strings, lists, dictionaries, sets, try/except, or recursion on this problem. If you do, you will receive a 0.

   We define a swappy number (a coined term) as a positive integer with at least three digits whose digits strictly alternate in parity (even, odd, even, odd, OR odd, even, odd, even). For example, 250, 327, and 8523 are swappy numbers, while 4112 is *not*, since it contains two adjacent digits with the same parity.

   Write the function `nthSwappyNumber(n)`, which returns the n^{th} swappy number, where n is a positive integer.

   Hint: You should decompose the problem using helper functions. In particular, it is strongly recommended that you first write a function `isSwappyNumber(a)` that determines whether a given integer a is a swappy number, and then use it to implement `nthSwappyNumber(n)`.

   Consider the following test cases:

```
# Invalid Cases
assert nthSwappyNumber(-4) == None # n <= 0
assert nthSwappyNumber("Hello") == None # n is not of type int

# Normal Cases
assert(nthSwappyNumber(1) == 101)
assert(nthSwappyNumber(3) == 105)
assert(nthSwappyNumber(5) == 109)
assert(nthSwappyNumber(6) == 121)
```

Extra space for Problem 2.

3. (7 points) **Free Response**: Digital Mirror Sum

   Do not use strings, lists, dictionaries, sets, try/except, or recursion on this problem. If you do, you will receive a 0.

   Write the function `digitalMirrorSum(n)` that takes a positive integer n and returns the sum of n and its mirror (the number with digits reversed). For example:

   - `digitalMirrorSum(123)` returns 444; since $123 + 321 = 444$
   - `digitalMirrorSum(500)` returns 505; since $500 + 005 = 500 + 5 = 505$
   - `digitalMirrorSum(1)` returns 2; since $1 + 1 = 2$

   Consider the following additional test cases:

```
assert(digitalMirrorSum(99) == 198) # 99 + 99
assert(digitalMirrorSum(1000) == 1001) # 1000 + 1
assert(digitalMirrorSum(4567) == 12221) # 4567 + 7654
```