

Name: \_\_\_\_\_ Andrew Id: \_\_\_\_\_

### 15-112 Spring 2025 Quiz 2

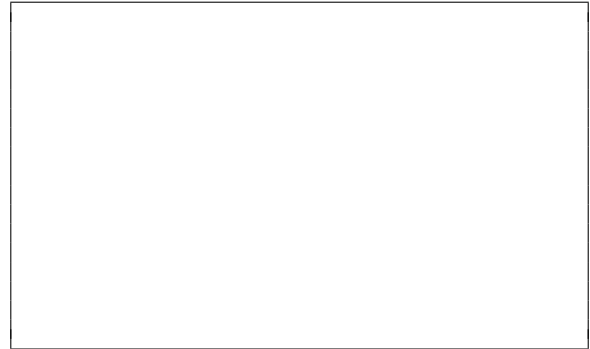
Up to 20 minutes. No calculators, no notes, no books, no computers. Show your work!  
Do not use strings, loops, lists, dictionaries, try/except, or recursion on this quiz.

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 g(x):  
    print('g', x, f(x))  
    if x > 10:  
        print('big x', x)  
    if type(x) == int:  
        x = x / 2  
    if type(x) == float:  
        x = x * 10  
    if x < 10:  
        print('small x', x)
```

```
def f(x):  
    print('f', x)  
    x *= 2  
    return x + 1
```

```
x = 4  
print(g(f(4)))  
print(x)
```



2. (6 points) **Free Response:** Crunch Encryption

Crunch Encryption encodes an integer by splitting each digit  $d$  into two parts:

- If the digit  $d$  is even, it splits into  $\frac{d}{2}, \frac{d}{2}$ .
- If the digit  $d$  is odd, it splits into  $\lfloor \frac{d}{2} \rfloor + 1, \lfloor \frac{d}{2} \rfloor$ .

The encryption preserves the sign of the number. For example:

- Encrypting 42 produces 2211:  $4 \rightarrow 22, 2 \rightarrow 11$ .
- Encrypting  $-37$  produces  $-2143$ :  $3 \rightarrow 21, 7 \rightarrow 43$ , with the negative sign unchanged.

Your task is to write a function `encryptCrunch(n)` that encrypts a given integer `n` using Crunch Encryption. You may assume `n` has at most 2 digits.

**Examples:**

```
assert encryptCrunch(42) == 2211
assert encryptCrunch(-37) == -2143
assert encryptCrunch(8) == 44
assert encryptCrunch(-5) == -32
assert encryptCrunch(0) == 0 # Zero remains unchanged (00 = 0)
assert encryptCrunch(1) == 10
```

**Notes:**

- In the description above,  $\lfloor x \rfloor$  is the mathematical notation for the floor of  $x$ .
- Do not use strings or loops to solve this problem.

3. (8 points) **Free Response:** Crunch Decryption

Now that you understand Crunch Encryption, let's reverse the process! Your task is to write a function `decryptCrunch(m)` that takes a value `m` and returns the original number before encryption. If `m` cannot be decrypted, return `None`.

**Examples:**

```
assert decryptCrunch(2211) == 42
assert decryptCrunch(-2143) == -37
assert decryptCrunch("hello") == None # Not an integer
assert decryptCrunch(87) == None # Invalid: 8+7 cannot be a digit
assert decryptCrunch(12) == None # Invalid split: 2 cannot be to the right of 1
assert decryptCrunch(21) == 3 # now it's ok 3 -> 2+1
assert decryptCrunch(2188) == None # cannot form valid digit from pair 88
assert decryptCrunch(4321) == 73 # Valid: 7 -> 43, 3 -> 21,
assert decryptCrunch(-1122) == -24 # Valid: 2 -> 11, 4 -> 22
```

**Notes:**

- When `m` is an integer, it will have at most 4 digits.
- Do not use strings or loops to solve this problem.