Name: _____ Andrew Id: _____

15-112 Fall 2025 Quiz 1

Up to 25 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. **Code Tracing**: Indicate what the following programs print. Place your answer (and nothing else) in the box next to the code.

   (a) (4 points) CT1

```
def ct1(x, y):
    x = x * 2 + y
    y = y // 2 + x
    print(x, y, y % x)
    z = x // 10 + y % 10
    print(z)

y = 3
x = 5
print(x, ct1(y, x), y)
```

   (b) (4 points) CT2

```
def ct2(a, b):
    if a + b > 10:
        print("High")
        return 1
    elif a < b:
        if b - a > 5:
            return 2
        else:
            b = b - 1
            print("Close")
    elif b % 2 == 0:
        print("Even")
        b = a
    if a == b and b % 2 == 0:
        print("Equal")
        return 6
    elif a != b or a * b > 0:
        print("Fine")
        return 4
    print("Done")
    return 5

a = 4
print(ct2(a - 1, a))
```

2. (6 points) **Free Response**: Point and Circle

   Write the function `pointAndCircle(px, py, cx, cy, r)` that determines the position of a point `(px, py)` relative to a circle centered at `(cx, cy)` with radius `r`. The arguments `px`, `py`, `cx`, `cy`, and `r` may be integers or floats, and you can assume `r` ≥ 0. The function must return:

   - `-1` if the point is strictly inside the circle,
   - `0` if the point is on the perimeter of the circle,
   - `1` if the point is outside the circle.

   For example, for a circle centered at $(0, 0)$ with radius 3:

   - `pointAndCircle(1, 1, 0, 0, 3)` returns `-1`, because $(1, 1)$ is strictly inside the circle.
   - `pointAndCircle(3, 0, 0, 0, 3)` returns `0`, because $(3, 0)$ is exactly on the perimeter.
   - `pointAndCircle(4, 0, 0, 0, 3)` returns `1`, because $(4, 0)$ lies outside the circle.

   More examples:

```
# Circle centered at (2, 2.5) with radius 2.5
assert pointAndCircle(2, 5.0, 2, 2.5, 2.5) == 0    # on the circle
assert pointAndCircle(3, 3, 2, 2.5, 2.5) == -1     # inside
assert pointAndCircle(-1.0, 2.5, 2, 2.5, 2.5) == 1 # outside

# other tests
assert pointAndCircle(1, 1, 1, 1, 3) == -1   # center is inside
assert pointAndCircle(42.0, 42.0, 42.0, 42.0, 0) == 0  # radius 0 means only center is "on"
```

3. (6 points) **Free Response**: Replace Digit

Write the function `replaceDigit(v, d1, d2)` that takes three integers `v`, `d1`, and `d2`, and behaves as follows:

- If $-999 \leq$ `v` ≤ 999, the function returns the integer formed by replacing every digit of `v` equal to `d1` with `d2`.
- If `v` is any other integer outside the range $[-999, 999]$, the function returns the string `"out of range"`.

You may assume that `d1` and `d2` are single digits in the range 0 through 9.

Examples:

```
assert replaceDigit(204, 0, 9) == 294  # replace 0 by 9 in 204 -> 294
assert replaceDigit(33, 3, 8) == 88    # replace 3 by 8 in 33 -> 88
assert replaceDigit(21, 5, 6) == 21    # no 5s to replace in 21 -> unchanged
assert replaceDigit(999, 9, 1) == 111  # replace 9 by 1 in 999 -> 111
assert replaceDigit(2025, 4, 7) == 'out of range' # 2025 too large -> out of range
assert replaceDigit(-113, 3, 2) == -112   # replace 3 by 2, and sign is preserved
assert replaceDigit(-999, 9, 0) == 0   # -999 -> -000 = 0
assert replaceDigit(1000, 4, 2) == 'out of range'  # just above valid range
assert replaceDigit(-1000, 4, 2) == 'out of range' # just below valid range
```

Notes:

- Do not use strings, loops, lists, or recursion to solve this problem. Use only integer arithmetic and conditionals.