1. (8.0 points)   What Would Python Display?

Assume the following code has been executed already.

```python
one = 1

def choose(one):
    if big(one):
        print('A')
        if huge(one):
            print('B')
    elif big(one) or huge(one):
        print('C')
    if big(one) or print('D'):
        print('E')
    else:
        print('F')

big = lambda x: x >= one
huge = lambda x: x > one

def which():
    one = 3
    def this():
        return one
    return one + 1
    return this
    one = 4
```

(a) **(6.0 pt)** Which lines are displayed by the interactive Python interpreter after evaluating `choose(one + one)`? **Select all that apply.**

☐ A

☐ B

☐ C

☐ D

☐ E

☐ None

☐ None of the above

(b) **(2.0 pt)** What is displayed by the interactive Python interpreter after evaluating `which()()`?

○ 2

○ 3

○ 4

○ 5

○ A function

○ An error occurs before anything is displayed

3. (8.0 points)   Nearly Square

Implement `near_square`, which takes positive integer **n** and non-negative integer **k**. It returns the largest integer less than or equal to **n** which is the product of two positive integers that differ by **k** or less. You may use `solve`, which is provided.

```python
def near_square(n, k):
    """Return the largest integer that is less than or equal to n and
    equals a * b for some positive integers a and b where abs(a - b) <= k.

    >>> near_square(125, 0)  # 11 * 11 = 121 and abs(11 - 11) = 0
    121
    >>> near_square(120, 3)  # 10 * 12 = 120 and abs(10 - 12) = 2
    120
    >>> near_square(120, 1)  # 10 * 11 = 110 and abs(10 - 11) = 1
    110
    """
    while True:

        gap = k

        while _____:
                 (a)
            x = _____
                  (b)
            if _____:    # Check if x is a whole number
                (c)

                return _____
                         (d)

            _____
             (e)

        _____
         (f)


def solve(b, c):
    """Returns the largest x for which x * (x + b) = c

    >>> solve(2, 120)  # x=10 solves x * (x + 2) = 120
    10.0
    >>> solve(2, 121)  # x=10.045... solves x * (x + 2) = 121
    10.045361017187261
    """
    return (b*b/4 + c) ** 0.5 - b/2
```

(a) (2.0 pt) Fill in blank (a). Select **all** that apply.

- ☐ gap
- ☐ gap != 0
- ☐ gap > 0
- ☐ gap >= 0