(1-3) An approximation for the square root of \( n \) can be generated using the following equation:

\[
x_{k+1} = \frac{1}{2} \left( x_k + \frac{n}{x_k} \right), \text{where } x_0 = 1
\]

Each value of \( x \) should be a better approximation for the square root of \( n \).

Given function \( \text{approx} \_\text{sqrt} \):

```python
def approx_sqrt(n, k):
    '''TYPE CONTRACT GOES HERE

    Generates an approximate square root of num, a positive integer, via an iterative process
    that runs iterations times. The approximate
    square root is returned.

    >>> approx_sqrt(1, 1)
    1.0
    >>> approx_sqrt(4, 1)
    2.5
    >>> approx_sqrt(4, 5)
    2.000000000000002
    '''
    value = 1
    for _ in range(k - 2):
        value = .5 * (value + n/value)
    return value
```

(1) supply the type contract that is consistent with the equation:

a) (int, int) \( \rightarrow \) float  
b) (float, float) \( \rightarrow \) int  
c) (int, int) \( \rightarrow \) None  
d) (str, int) \( \rightarrow \) float  

(2) Replace ??-1 with the code needed to implement the approximation.

a) .5  
b) 1  
c) k  
d) n  
e) value  

(3) Replace ??-2 with the code needed to implement the approximation.

a) .5  
b) 1  
c) k  
d) n  
e) value  

(4) The decimal representation of binary 1111 is

a) 1  
b) 15  
c) 16  
d) 17  
e) no decimal equivalent
(5) What is the result of executing the following Python code:

```python
>>> for bit in '10':
    if bit not in '01':
        print(False)
-1
```

Replace -1 with the result:

a) False  b) True  c) False  c) '01'  e) nothing is printed

(6-7) Given the following Python code:

```python
>>> x = 'CIS 210'
>>> id(x)
4391509160
>>> y = x
>>> id(y)
-1
>>> x = 'the end'
>>> id(x)
-2
```

(6) 4391509160 refers to a(n)

a) hexadecimal number  b) function  c) None type  d) keyword  e) memory location

(7) The value printed at -1 will also be 4391509160 (yes or no); the value printed at -2 will also be 4391509160 (yes or no).

a) yes/yes  b) no/no  c) yes/no  d) no/yes

(8-9)

(8) What will be printed when the following Python code is executed?

```python
n = 5
mysum = 0
for ctr in range(1, n):
    mysum += ctr
print(mysum)
-1
```

(8) Replace -1 with the result:

a) 0  b) 5  c) 10  d) 15  e) None

(9) This code is an example of

a) accumulator pattern  b) TypeError  c) conditional  d) indefinite iteration  e) infinite loop
(10) Given the following Python code:

```python
>>> import math
>>> from math import pi
>>> dir()
```

Which of the following would you expect to see in the __main__ namespace after `dir()` is executed?

a) `import`  
b) `math`  
c) `sqrt`  
d) `math.pi`  
e) `math.sqrt`

(11-12) Given the following Python code:

```python
0 import math
1
2 def isInCircle(x, y):
3     '''(number, number) -> ??
4     Quiz.
5     '''
6     d = math.sqrt(x**2 + y**2)
7     isIn = (d <= 1)
8     return isIn
```

(11) Complete the type contract:

a) `int`  
b) `float`  
c) `number`  
d) `bool`  
e) `str`

(12) Indicate which lines of code would need to be changed for `isInCircle` to check whether point `(x, y)` were inside a circle with a radius of any length.

a) 0, 9  
b) 2, 3, 8  
c) 2, 8, 9  
d) 2, 8  
e) 7, 8, 9

(13-15) Given the following Python code:

```python
def quadruple(x):
    ''' quiz '''
    y = 4
    result = y * x
    return result

>>> x = 5
>>> quadruple(10)
??-1
>>> x
??-2
>>> y
??-3
```
(13) Replace ??-1 with the expected result:

a) 5  b) 10  c) 20  d) 40  e) NameError

(14) Replace ??-2 with the expected result:

a) 5  b) 10  c) 20  d) 40  e) NameError

(15) Replace ??-3 with the expected result:

a) 4  b) 16  c) 10  d) 40  e) NameError

(16) Given the following Python code:

```python
0  def isDivisible(m, n):
1      ''' quiz '''
2      return m % n == 0
4  def higherLevel(m, n):
5      ''' quiz '''
7      if isDivisible(m, n):
8          print('yes')
9      else:
10         print('no')
12      return None
```

Which of the following lines of code, if substituted for line 8, would affect the result of executing `>>> isDivisible(7, 2)`

a) if isDivisible:

b) if isDivisible(m, n) == True:

c) isDiv = isDivisible(m, n)
   if isDiv:

 d) a) and b)

e) none of these changes would affect the result