1. What are the contents of \( l_2 \) after the following code is executed?

```python
l1 = ['one', 'two', 'three']
l2 = [(l1[i], l1[i+1]) for i in range(len(l1)-1)]
```

(a) []
(b) [('one', 'three')]
(c) [('two', 'one'), ('three', 'two')]
(d) [('one', 'two'), ('two', 'three')]

2. What is the value of \( a \) after the following code is executed?

```python
a = 1
b = 2
for _ in range(5):
    a, b = b, 2*b
```

(a) 1
(b) 25
(c) 32
(d) 64

3. What is the value of the expression \( \text{dct}['fish'] \) after the following code is executed?

```python
words = ['one', 'fish', 'two', 'fish', 'red', 'fish', 'blue', 'fish']
dct = {}
for i in range(len(words)):
    w = words[i]
    if w in dct:
        dct[w][0] += 1
        dct[w][1] = i
    else:
        dct[w] = [1, i]
```

(a) [7, 4]
(b) [4, 7]
(c) [4, 1]
(d) [9, 4]
4. For this and the next problem, consider the following function definition:

```python
def foo(a, b=5, c=10):
    return a+b+c
```

What is the return value of the call `foo(1, 2, 3)`?

(a) 15  
(b) 6  
(c) 5  
(d) 16

5. What is the return value of the call `foo(2, 3)`?

(a) 15  
(b) 5  
(c) 2  
(d) 8

6. Consider the following function definition:

```python
def bar(*x):
    res = x[0]
    for x in x[1:]:
        res += x
    return res
```

What is the return value of the call `bar([1, 2], [4, 5], [9])`?

(a) 14  
(b) [1, 4, 9]  
(c) [1, 2, 4, 5, 9]  
(d) [14, 7]
7. Consider the following class definition and subsequent code:

```python
class Foo:
    def __init__(self, name):
        global count
        self.identity = name + str(count)
        count += 1

count = 0
f1 = Foo('widget')
f2 = Foo('sprocket')
```

What is the value of the tuple `(f1.identity, f2.identity)`?

(a) (`'widget', 'sprocket'`)  
(b) (`'widgetzero', 'sprocketone'`)  
(c) (`'sprocket1', 'sprocket2'`)  
(d) (`'widget0', 'sprocket1'`)  

8. Consider the following class definition and subsequent code:

```python
class Bar:
    def __init__(self):
        self.total = 0

    def __getitem__(self, key):
        self.total -= key
        return self.total

    def __setitem__(self, key, val):
        self.total += val

    def __len__(self):
        return self.total

b = Bar()
b[0] = 20
b[10] = 30
val = b[10]
```

What is the value of the expression `len(b)`?

(a) **40**  
(b) **50**  
(c) **60**  
(d) **10**
9. Given that iterable is an iterable object, which of the following emulates the behavior of a for loop to iterate over its contents?

(a) it = iterable
    while True:
        i = iter(it)
        x = next(i)
        # do something with x
        if not i:
            break

(b) it = iter(iterable)
    while True:
        try:
            x = next(it)
            # do something with x
        except StopIteration:
            break

(c) it = next(iterable)
    while True:
        try:
            x = iter(it)
            # do something with x
        except StopIteration:
            break

(d) it = iter(iterable)
    while True:
        x = next(it)
        # do something with x
        else:
            raise StopIteration
10. Consider the following class definition and subsequent code:

```python
class MyIter:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __iter__(self):
        return self

    def __next__(self):
        if self.x > self.y:
            raise StopIteration
        else:
            ret = self.x
            self.x *= 2
            return ret
```

```python
l = []
for x in MyIter(3, 50):
    l.append(x)
```

What are the contents of the list `l`?

(a) `[3, 50]`

(b) `[3, 6, 12, 24, 48]`

(c) `[53]`

(d) `[6, 12, 18, 24, 30, 36, 42, 48]`
11. What is the worst-case run-time complexity of locating and retrieving the element in middle position (by index) of an array-backed list of N elements?

(a) \(O(1)\)
(b) \(O(\log N)\)
(c) \(O(N)\)
(d) \(O(N^2)\)

12. What is the worst-case run-time complexity of counting the number of times a given value occurs in an unsorted, array-backed list of N elements?

(a) \(O(1)\)
(b) \(O(\log N)\)
(c) \(O(N)\)
(d) \(O(N^2)\)

13. What is the worst-case run-time complexity of using binary search to determine whether a given value exists in a sorted, array-backed list of N elements?

(a) \(O(1)\)
(b) \(O(\log N)\)
(c) \(O(N)\)
(d) \(O(N^2)\)

14. What is the worst-case run-time complexity of extending an array-backed list with the contents of another list containing N elements?

(a) \(O(1)\)
(b) \(O(\log N)\)
(c) \(O(N)\)
(d) \(O(N^2)\)

15. What is the worst-case run-time complexity of removing an arbitrary element from an array-backed list of N elements?

(a) \(O(1)\)
(b) \(O(\log N)\)
(c) \(O(N)\)
(d) \(O(N^2)\)
16. Which of the plots best depicts the worst-case run-time complexity of the following function?

```python
def f(n):
    res = 0
    while n > 0:
        res += n
        n = n // 10
    return res
```

(a)  
(b)  
(c)  
(d)
Which of the plots best depicts the worst-case run-time complexity of the following function?

```python
def f(lst):  # `lst` is a Python list
    n = len(lst) - 1
    while n >= 0:
        for i in range(len(lst)):
            if i != n:
                lst[i] *= lst[n]
        n -= 1
    return lst
```

(a) ![Graph A](image)
(b) ![Graph B](image)
(c) ![Graph C](image)
(d) ![Graph D](image)
18. Which of the plots best depicts the worst-case run-time complexity of the following function?

def f(lst): # `lst` is a Python list
    res = 0
    step = len(lst) // 10
    for i in range(0, len(lst), step):
        res += lst[i]
    return res

(a)  
(b)  
(c)  
(d)
19. Which of the plots best depicts the worst-case run-time complexity of the following function?

```python
def f(n):
    res = 0
    for i in range(n, 0, -1):
        res += i
    return res
```

[Diagram of plots (a), (b), (c), (d)]
20. Which snippet correctly completes the implementation of insertion sort on a list?

```python
def insertion_sort(lst):
    for i in range(1, len(lst)):
        for j in range(i, 0, -1):
            if lst[j-1] > lst[j]:
                lst[j-1], lst[j] = lst[j], lst[j-1]

        for j in range(i, 1, -1):
            if lst[j-1] > lst[j+1]:
                lst[j-1], lst[j+1] = lst[j+1], lst[j-1]

        for j in range(i):
            if lst[i] > lst[j]:
                lst[i] = lst[j]

        for j in range(1, i+1):
            if lst[j] > lst[i]:
                lst[j] = lst[i]
```

(a) `for j in range(i, 0, -1):`
    `if lst[j-1] > lst[j]:`
    `lst[j-1], lst[j] = lst[j], lst[j-1]`

(b) `for j in range(i, 1, -1):`
    `if lst[j-1] > lst[j+1]:`
    `lst[j-1], lst[j+1] = lst[j+1], lst[j-1]`

(c) `for j in range(i):`
    `if lst[i] > lst[j]:`
    `lst[i] = lst[j]`

(d) `for j in range(1, i+1):`
    `if lst[j] > lst[i]:`
    `lst[j] = lst[i]`

21. Which snippet correctly completes the implementation of `__delitem__` in an array-backed list (assuming a valid index ≥ 0)?

```python
def __delitem__(self, idx):
    for i in ________________________:
        self.data[i-1] = self.data[i]
    del self.data[len(self.data)-1]
```

(a) `range(idx)`

(b) `range(len(self.data))`

(c) `range(idx+1, len(self.data))`

(d) `range(len(self.data), idx, -1)`
22. Which snippet correctly completes the implementation of `__iter__` in an array-backed list?

```python
def __iter__(self):
    ------------------------------
    ------------------------------
    ------------------------------
(a) for x in self:
    yield x
(b) while iter(self):
    yield next(self)
    raise StopIteration
(c) for i in range(len(self)):
    return self.data[i]
(d) for i in range(len(self)):
    yield self.data[i]
```

23. Which snippet correctly implements `pop` in an array-backed list, given a working `__delitem__` method and that `self.data` is a `ConstrainedList` (as in lab)?

```python
def pop(self, idx=-1):
    ------------------------------
    ------------------------------
    return val
(a) val = self.data.pop(idx)
(b) val = self[idx]
    self.remove(val)
(c) val = self.data[idx]
    del self.data[idx]
(d) val = self[idx]
    del self[idx]
```
24. Which snippet correctly implements `splice` in an array-backed list, which inserts the contents of the provided list argument (`lst`) into the array list starting at position `idx`?

```python
def splice(self, idx, lst):
    m = len(self)
    n = len(lst)
    for _ in range(n):
        self.data.append(None)

    (a) for i in range(m+n, m, -1):
        self.data[i-1] = self.data[i-n-1]
    for i in range(n):
        self.data[idx+i] = lst[i]

    (b) for i in range(m, idx, -1):
        self.data[i+n-1] = self.data[i-1]
    for i in range(n):
        self.data[idx+i] = lst[i]

    (c) for i in range(idx, m+n):
        self.data[m+n-1] = self.data[m-1]
    for i in range(idx, idx+n):
        self.data[i] = lst[i]

    (d) for i in range(idx, m+n):
        self.data[m+n-i-1] = self.data[m-i-1]
    for i in range(idx, m):
        self.data[i] = lst[i]
```

E.g., calling `splice` with `idx=2` and `lst=['one', 'two', 'three']` on an ArrayList which currently contains `[1, 2, 3, 4, 5]` should result in `[1, 2, 'one', 'two', 'three', 3, 4, 5]`