1. What exactly is []?
   ANSWER: The empty list value, which is a list value that contains no items.

2. In a list of values stored in a variable called spam, how would you assign the value 'hello' as the third value? (Assume [2, 4, 6, 8, 10] are in spam.)
   ANSWER:

```
# 3rd value in the list will be at index 2.
spam=[2, 4, 6, 8, 10]
spam[2]='hello'
print(spam)
```

```
[2, 4, 'hello', 8, 10]
```

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

3. What is the value of spam[int(int('3' * 2) / 11)]?
   ANSWER: d, refer below snap

```
In [5]: spam=['a','b','c','d']
        print(spam[int(int('3'*2)/11)])

        d
```

4. What is the value of spam[-1]?
   ANSWER: d, refer below snap

```
spam=['a','b','c','d']
print(spam[-1])
```

```
d
```

5. What is the value of spam[:2]?
   ANSWER: ['a', 'b']

```
spam=['a','b','c','d']
print(spam[:2])
```

```
['a', 'b']
```

Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.

6. What is the value of bacon.index('cat')?
   ANSWER: 1

```
In [ ]: bacon = [3.14, 'cat', 11, 'cat', True]
        bacon.index('cat') # it returns the index of first occurrence of 'cat'

Out[ ]: 1
```

7. How does bacon.append(99) change the look of the list value in bacon?
   ANSWER:

```
In [13]: bacon=[3.14,'cat',11,'cat',True]
         bacon.append(99)
         print(bacon)

         [3.14, 'cat', 11, 'cat', True, 99]
```

8. How does bacon.remove('cat') change the look of the list in bacon?
   ANSWER:

```
bacon=[3.14,'cat',11,'cat',True]
bacon.remove('cat')
print(bacon)

[3.14, 11, 'cat', True]
```

9. What are the list concatenation and list replication operators?
   ANSWER:

```
l1=[1,2,3]
l2=[4,5,6]
l3=l1+l2  # list concatenation operator
print(l3)

l1=[1,2,3]
l2=l1*3    #list replication operator
print(l2)

[1, 2, 3, 4, 5, 6]
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

10. What is difference between the list methods append() and insert()?
    ANSWER: append() : appends the object at the end of the list.
    Insert(): inserts the object at the defined index value.

```
l1=[3,4,5]
l2=[7,9,10]
l1.append(6)
print(l1)
l2.insert(1,8)
print(l2)
```

```
[3, 4, 5, 6]
[7, 8, 9, 10]
```

11. What are the two methods for removing items from a list?
    ANSWER: pop() and remove() are the two methods to remove the items from list.

```
In [22]: l1=[3,4,5,6,7,8]
         l1.pop()
         print(l1)
         l1.remove(4)
         print(l1)
```

```
[3, 4, 5, 6, 7]
[3, 5, 6, 7]
```

12. Describe how list values and string values are identical.
    ANSWER:
    - Can be used in for loops
    - Both lists and strings can be passed to len()
    - Have indexes and slices
    - Can be used with the in and not in operators
    - Can be concatenated or replicated

13. What's the difference between tuples and lists?
    ANSWER: List is mutable while tuples are immutable. List is defined using [] while tuples are defined using ().

14. How do you type a tuple value that only contains the integer 42?
    ANSWER:

```
In [24]: t1=(42,)
         print(t1)

         (42,)
```

15. How do you get a list value's tuple form? How do you get a tuple value's list form?
    ANSWER:

```
In [26]: l1=[1,2,3]
         print(tuple(l1)) # converting list to tuple

         (1, 2, 3)

In [28]: t1=[4,5,6]
         print(list(t1))  # converting tuple to list

         [4, 5, 6]
```

16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?
    ANSWER: Variables will contain references to list values rather than list values themselves.

17. How do you distinguish between copy.copy() and copy.deepcopy()?

    ANSWER: The copy.copy() function will do a shallow copy of a list, while the copy.deepcopy() function will do a deep copy of a list. That is, only copy.deepcopy() will duplicate any lists inside the list.