1. What exactly is []?
   * [] – denotes an empty list.
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.)
   * spam[3]="hello"

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

1. What is the value of spam[int(int('3' \* 2) / 11)]?
   * 'd'
2. What is the value of spam[-1]?
   * ‘d’
3. What is the value of spam[:2]?
   * ['a', 'b']

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

1. What is the value of bacon.index('cat')?
   * 1
2. How does bacon.append(99) change the look of the list value in bacon?
   * [3.14, 'cat', 11, 'cat', True, 99]
3. How does bacon.remove('cat') change the look of the list in bacon?
   * [3.14, 11, 'cat', True, 99]
4. What are the list concatenation and list replication operators?
   * + is the list concatenation operator
   * \* is the list replicate operator
5. What is difference between the list methods append() and insert()?
   * append() adds the specific element always at the last index while insert() function adds specific element to a specific index of the list
6. What are the two methods for removing items from a list?
   * remove() and pop() shall be used to remove
7. Describe how list values and string values are identical.
   * List and strings are similar in a way that both contain ordered collection of data in a sequence.
8. What's the difference between tuples and lists?
   * Tuples are immutable while elements in a list are mutable
9. How do you type a tuple value that only contains the integer 42?
   * (42)
10. How do you get a list value's tuple form? How do you get a tuple value's list form?
    * List 🡪 tuple : list(tuple\_t)
    * Tuple 🡪 list : tuple(list\_l)
11. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?
    * variable contain references to list values
12. How do you distinguish between copy.copy() and copy.deepcopy()?
    * copy.copy() creates a new object which stores the reference of the original elements. However it doesn’t create a copy of nested objects, instead it just copies the reference of nested objects. Hence if the nested list of parent list is changed it gets changed in the newly created copy too.
    * Whereas copy.deepcopy() create a copy of nested objects. Thus deep copy creates independent copy of original object and all its nested objects. Hence if the nested list of parent list is changed it does not affects or changes the newly created copy.