CSE160 Practice Problems: Sorting

1. **Which of the following statements about `sorted` and `sort` is TRUE?**
   A. The `sorted()` function sorts a list in place and returns `None`.
   B. The `sort()` method returns a sorted version of a list without modifying the original.
   C. The `sorted()` function returns a new sorted list and doesn't modify the original list.
   D. The `sort()` method can be called on any iterable, not just lists.

2. What will happen with the following code?

   fruits = ["apple", "banana", "cherry"]

   print(sorted(fruits, key=len))

   A. It sorts `fruits` alphabetically by default.

   B. It sorts `fruits` in reverse alphabetical order.

   C. It sorts `fruits` based on the length of each item.

   D. It raises an error because `len` is not a valid sorting key.

3. Given the list of tuples representing students and their scores:

   students = [("Alice", 88), ("Bob", 92), ("Charlie", 88), ("David", 85)]

   Which of the following sorts `students` by score in ascending order, keeping the original order for students with the same score?

   A. `sorted(students, key=itemgetter(1), reverse=True)`

   B. `sorted(students, key=itemgetter(1))`

   C. `students.sort(key=itemgetter(0))`

   D. `students.sort()`

4. ***Tuple Problem*: Accessing and Manipulating Tuples**

   Given the following tuple representing a book's details:

   `book = ("The Great Gatsby", "F. Scott Fitzgerald", 1925)`

   Which of the following statements will correctly extract only the book title from `book`?

   A. `title = book[1]`

   B. `title = book[0]`

   C. `title = book[2]`

   D. `title = book[3]`

   What will happen if you try to execute the following line?

   `book[2] = 1926`

   A. The book tuple will be updated to `("The Great Gatsby", "F. Scott Fitzgerald", 1926)`.

   B. A `TypeError` will be raised because tuples are immutable.

   C. The book's publication year will be changed to 1926 without any error.

   D. The code will execute but will not make any change to `book`.

   Which of the following will create a new tuple with an added genre field?

   `genre = "Novel"`

   A. `book += genre`

   B. `book = book + genre`

   C. `book = book + (genre,)`

   D. `book = (genre,) + book`

5. **Problem: Product Inventory Management (Hand Coding)**

   You are managing an inventory of products in a warehouse. Each product has a name and a quantity in stock. The information is stored in a list of tuples:

   ```
   inventory = [("Apples", 150), ("Bananas", 80), ("Oranges", 150), ("Pears", 200)]
   ```

   Each tuple contains:

   - The product name (a string)
   - The quantity in stock (an integer)

Task 1: Write code to **sort** the `inventory` list by quantity in ascending order, so that products with the lowest stock appear first. If two products have the same quantity, they should remain in their original order.

- **Expected Output:**

  ```
  [('Bananas', 80), ('Apples', 150), ('Oranges', 150), ('Pears', 200)]
  ```

Task 2: Write code to **sort** the `inventory` list by quantity in descending order, so that products with the highest stock appear first. If two products have the same quantity, they should be sorted alphabetically by name.

- **Expected Output:**

  ```
  [('Pears', 200), ('Apples', 150), ('Oranges', 150), ('Bananas', 80)]
  ```

Answers:

1. C

2. C

3. B

4. B B C

5. Example Solution:

```python
from operator import itemgetter

# Task 1: Sort by Quantity (Ascending)
sorted_inventory_asc = sorted(inventory, key=itemgetter(1))
print("Ascending by Quantity:", sorted_inventory_asc)

# Task 2: Sort by Quantity (Descending), then by Name
sorted_inventory_desc_name = sorted(inventory, key=itemgetter(1, 0), reverse=True)
print("Descending by Quantity, then Name:", sorted_inventory_desc_name)
```