CSPP Remedial

Time: 2 hours Score: 25 Points

1. **Description: (4 Points)**
   Write a Python function `find_uniques(nums)` that takes a list of integers `nums` and returns a new list containing only the elements that appear exactly once in the original list. The elements in the returned list should be in the order they first appeared in the input list.

   Test Cases:
   assert find_uniques([1, 2, 2, 3, 4, 4, 5]) == [1, 3, 5]
   assert find_uniques([7, 7, 7, 7]) == []
   assert find_uniques([]) == []
   assert find_uniques([2, 2, 3, 3, 4, 4, 4, 5]) == [5]

2. **Description: (5 Points)**
   Implement a Python function merge_dicts(d1, d2) that merges two dictionaries into a new dictionary. If a key exists in both dictionaries, the value from d2 should overwrite the value from d1. Return the new dictionary.

   Test Cases:
   assert merge_dicts({'a': 1, 'b': 2}, {'b': 3, 'c': 4}) == {'a': 1, 'b': 3, 'c': 4}
   assert merge_dicts({}, {'a': 10, 'b': 20}) == {'a': 10, 'b': 20}
   assert merge_dicts({'x': 5}, {}) == {'x': 5}
   assert merge_dicts({'k': 7}, {'k': 8, 'j': 9}) == {'k': 8, 'j': 9}

3. **Description: (5 Points)**
   Create a Python function longest_consecutive(nums) that finds and returns the length of the longest consecutive elements sequence in an unsorted list of integers.

   Test Cases:
   assert longest_consecutive([100, 4, 200, 1, 3, 2]) == 4
   assert longest_consecutive([1, 9, 3, 10, 4, 20, 2]) == 4
   assert longest_consecutive([]) == 0
   assert longest_consecutive([50]) == 1

Explanation:
Input: [100, 4, 200, 1, 3, 2]
The consecutive sequences in this list are [1, 2, 3, 4] and [100] and [200].
The longest sequence here is [1, 2, 3, 4], which contains 4 numbers.
Hence, the output is 4.

4. **Description: (5 Points)**
   Implement a function max_profit(prices) that calculates the maximum profit that could be made by buying and then selling a single share of stock on different days. Return the profit.

   Test Cases:
   assert max_profit([7, 1, 5, 3, 6, 4]) == 5
   assert max_profit([7, 6, 4, 3, 1]) == 0
   assert max_profit([1, 2, 3, 4, 5]) == 4
   assert max_profit([1, 6, 2, 8, 3, 10]) == 9

5. **Description: (6 Points)**
   Write a function can_partition(nums) that returns True if and only if you can partition the array into two subsets such that the sum of elements in both subsets is the same.

   Test Cases:
   assert can_partition([1, 5, 11, 5]) == True
   assert can_partition([1, 2, 3, 5]) == False
   assert can_partition([1, 1]) == True
   assert can_partition([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]) == True