CPSC 250 – Programming for Data Manipulation – Test 2

Time: 75 minutes
Instructions: Answer all questions on this test paper. No electronic devices or notes are permitted. Be sure to write clearly.

Part A: Multiple Choice (10 points, 2 points each)

Circle the one correct answer.

1. What is the result of modifying an element of a list passed to a function?
   A. It raises a TypeError.
   B. Only a local copy is changed.
   C. The original list is changed.
   D. The function cannot access the list.

2. What is the purpose of the `__init__` method in a Python class?
   A. It defines a string representation of the object.
   B. It is used to compare objects.
   C. It initializes the object's state.
   D. It deletes the object.

3. What does Python's garbage collector do?
   A. Prevents memory leaks by closing files.
   B. Automatically deletes unused variables after each loop.
   C. Frees memory by deleting objects with zero references.
   D. Ensures all objects are copied when passed to functions.

4. What is one advantage of using setter methods instead of direct attribute access?
   A. Setter methods reduce the number of attributes in a class.
   B. Setter methods automatically call `__str__()` when an attribute changes.
   C. Setter methods can include validation logic before updating a value.
   D. Setter methods are faster than direct access.

5. Which of the following correctly overloads the equality (`==`) operator in a class?
   A. `def __eq__(x, y):`
   B. `def __equal__(self, other):`
   C. `def __compare__(self, other):`
   D. `def __eq__(self, other):`

Part B: Find the Error (15 points, 3 points each)

Each of the following code snippets has one or more errors. Identify and explain them.

1.
```
def double(n):
    n *= 2

value = 7
result = double(value)
print(result)
```

2.
```
class Student:
    def __init__(self, name, grade):
        name = name
        grade = grade
```

3.
```
def add_name(names=None):
    names.append("Alice")
    return names
```

4.
```
class Counter:
    def __init__(self):
        self.count = 0

    def __str__():
        return f"Count is {self.count}"
```

5.
```
class Vector:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, v):
        return (self.x + v.x, self.y + v.y)
```

Part C: Code Writing (30 points)

1. (10 points) Write a function `modify_list(mylist)` that appends the value `42` to the list. Call the function on a list from main code, and explain why the original list is or is not changed.

2. (10 points) Write a class called `Book` that has:

   - Three instance variables: `title`, `author`, and `pages`
   - A constructor
   - Getters and setters for all variables
   - A method `summary()` that returns a description string
   - A `__str__()` method that returns something like `"Book: Title by Author (300 pages)"`

3. (10 points) Modify your `Book` class to support the `+` operator to combine two books into a new book:

   - Title = "Collection"
   - Author = "Various"
   - Pages = sum of both

Part D: Code Commentary (20 points)

The following program creates a class representing an inventory item. **Write comments next to each line** explaining what it does.

```
class InventoryItem:

    def __init__(self, name, quantity):
        self.name = name
        self.quantity = quantity


    def restock(self, amount):
        self.quantity += amount


    def sell(self, amount):
        if amount > self.quantity:
            print("Not enough in stock")
        else:
            self.quantity -= amount


    def __str__(self):
        return f"{self.name}: {self.quantity} in stock"
```