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 happens when you pass a mutable object like a list to a Python function?
   A. A new copy is created
   B. It behaves like pass by value
   C. The function can modify the original list
   D. The function cannot access the list

2. Which of the following best describes what `__str__()` is used for?
   A. Converts all strings in a class to uppercase
   B. Returns a string representation of an object
   C. Initializes an object
   D. Deletes a string from memory

3. In Python, when does garbage collection typically occur?
   A. When the CPU is idle
   B. When a variable is reassigned
   C. When an object's reference count reaches zero
   D. At the end of a function call

4. What is the purpose of a setter method?
   A. To display object attributes
   B. To initialize instance variables
   C. To allow controlled access to modify private variables
   D. To overload arithmetic operators

5. Which statement about operator overloading is true?
   A. You cannot redefine `+` in Python
   B. Operator overloading is only for strings
   C. You use special methods like `__add__()` to define operator behavior
   D. Operator overloading requires C++

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 change_value(x):
    x = x + 1

y = 10
change_value(y)
print(y)  # Expected output: 11
```

2.
```
class Circle:
    def __init__(radius):
        self.radius = radius
```

3.
```
def append_item(mylist=[]):
    mylist.append(1)
    return mylist
```

4.
```
class Dog:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return self.name

buddy = Dog("Buddy")
print(buddy.__str__)
```

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

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

Part C: Code Writing (30 points)

1. (10 points) Write a function `swap_values(a, b)` that attempts to swap two variables. Then write a short explanation as to **why or why not** the values change outside the function.

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

   - Two instance variables: `width` and `height`
   - A constructor
   - Getters and setters for both variables
   - A method `area()` that returns the area
   - A `__str__()` method that returns a string like `"Rectangle(width=3, height=4)"`

3. (10 points) Extend your `Rectangle` class to allow adding two rectangles using `+` so that `r3 = r1 + r2` creates a new rectangle with combined width and height.

Part D: Code Commentary (20 points)

The following program creates a class representing a simple bank account. **Write comments next to each line** explaining what it does.

```
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            print("Insufficient funds")

    def __str__(self):
        return f"{self.owner}'s account balance: ${self.balance}"
```