Case Study 2: Concurrent Map Update with Lambda

Total marks = 20 marks

You are developing a multi-threaded analytics system for a university portal that tracks student activity. Each time a student logs in, the system should increment their login count in a shared map. To handle high concurrency and ensure data consistency without using explicit synchronization, you decide to use a `ConcurrentHashMap` along with lambda expressions for atomic updates. You also want to ensure that multiple threads can update the same student's login count safely and efficiently. The sample coding is shown below.

```java
package org.example.Week_13;

import java.util.concurrent.ConcurrentHashMap;

public class StudentLoginTracker {
    public static void main(String[] args) {
        ConcurrentHashMap<String, Integer> loginCounts = new ConcurrentHashMap<>();
        loginCounts.put("student123", 0);

        Runnable incrementLogin = () -> {
            for (int i = 0; i < 500; i++) {
                loginCounts.compute( key: "student123", ( String k, Integer v) -> v + 1);
            }
        };

        Thread t1 = new Thread(incrementLogin);
        Thread t2 = new Thread(incrementLogin);

        t1.start();
        t2.start();

        try {
            t1.join();
            t2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("Final login count for student123: " + loginCounts.get("student123"));
    }
}
```

Answer the questions based on the case study.

1. Why is ConcurrentHashMap used instead of HashMap in this case?

(2 Marks)

2. What is the role of the lambda expression inside the compute() method?

(2 Marks)

3. How does this approach ensure thread safety without using synchronized blocks?

(3 Marks)

4. What will be the expected final login count for student123, and briefly explain why?

(3 Marks)

5. If you added a third thread that also executes incrementLogin, what would the new expected final login count be? Justify your answer.

(4 Marks)

6. Modified Code for Two Students (student123 and student456):

(6 Marks)