Assignment 4

The goal of this assignment is to understand and compare three automated unit test generation approaches—**EvoSuite, Randoop, and ChatGPT**—in terms of their effectiveness, strengths, and limitations.

You need to perform the following steps **for the Java code mentioned below**:

1. Use **EvoSuite** to automatically generate unit tests for the project.
    - EvoSuite is a search-based test generation tool that automatically creates JUnit test cases using evolutionary algorithms to maximize code coverage.
    - When you run EvoSuite on your project, it analyzes the Stack.java file and generates multiple test classes that thoroughly test its methods.
    - The output usually includes files named EvoSuiteTest.java and EvoSuiteRegressionTest.java, which contain automatically generated unit tests and regression tests.
    - These files must be placed inside the test/evosuite/ folder of your project repository.
    - Sample command: java -jar evosuite.jar -class Stack -projectCP target/classes
2. Use **Randoop** to do the same.
    - Randoop is a random test generator that produces sequences of method calls to find unexpected program behaviors and exceptions.
    - It generates JUnit test files that test different method combinations in your Stack.java class.
    - The output generally consists of RegressionTest0.java and ErrorTest0.java, where the regression tests check for consistent program behavior and the error tests record any exceptions found.
    - These files should be saved inside the test/randoop/ folder in your repository.
    - Sample command: java -classpath .:randoop-all-4.3.2.jar randoop.main.Main gentests --testclass=Stack
3. Use **ChatGPT** to generate a set of unit test cases for the same project.
    - ChatGPT can be used to create human-written unit tests by prompting it to generate JUnit test cases for Stack.java.
    - The generated tests are typically cleaner and more readable, focusing on edge cases, logic correctness, and expected outputs.
    - The test file created through this method should be named ChatGPTTest.java and placed inside the test/chatgpt/ folder in your repository.
    - Sample prompt: “Generate JUnit test cases for the following Java class Stack.java that test all methods and edge cases.”

- Run all three sets of generated tests.
- Measure and report code coverage (statement, branch, and method coverage).
- You may use tools such as **JaCoCo, EclEmma, Clover or IntelliJ** coverage reports.
- Deadline for the assignment is **2nd November, 2025 11:59 PM**.

Teamwork

- Students must work in a team of **3.**
- The team members are NOT allowed to be from the same PROJECT GROUP.
- Each team is responsible for executing all three methods of test generation and can **create a joint report together**, but **each student must submit the report individually** for evaluation.
- Each project report must have names and NetID of each member at the start.

Software Code "Stack.java"

Use the following Java class as the subject under test:

```
public class Stack {
    private int[] stack;
    private int top;
    private int capacity;

    // Constructor
    public Stack(int size) {
        capacity = size;
        stack = new int[size];
        top = -1;
    }

    // Push an element
    public void push(int value) {
        if (top < capacity - 1) {
            stack[++top] = value;
        } else {
            throw new IllegalStateException("Stack is full");
        }
    }

    // Pop an element
    public int pop() {
        if (top >= 0) {
            return stack[top--];
        } else {
            throw new IllegalStateException("Stack is empty");
```

```
        }
    }

    // Peek top element
    public int peek() {
        if (top >= 0) {
            return stack[top];
        } else {
            throw new IllegalStateException("Stack is empty");
        }
    }

    // Check if stack is empty
    public boolean isEmpty() {
        return top == -1;
    }

    // Check if stack is full
    public boolean isFull() {
        return top == capacity - 1;
    }

    // Return current size
    public int size() {
        return top + 1;
    }

    public static void main(String[] args) {
        Stack s = new Stack(5);
        s.push(10);
        s.push(20);
        s.push(30);
        System.out.println("Top element: " + s.peek());
        System.out.println("Popped: " + s.pop());
        System.out.println("Stack size: " + s.size());
        System.out.println("Is empty: " + s.isEmpty());
    }
}
```

Report

Prepare a short report (2–3 pages) that includes the following sections:

1. **Strengths of EvoSuite:** Discuss what EvoSuite does well—such as search-based generation, assertions, mutation resistance, etc.

2. **Strengths of Randoop:** Highlight its advantages—such as simplicity, random exploration, or speed.

3**. Strengths of ChatGPT-based Test Generation:** Discuss how ChatGPT-generated tests differ, their readability, creativity, and accuracy.

4. **Process and Prompts of Chatgpt**: Discuss and mention how you used ChatGPT to generate the testcases.

5. **Comparison:** Explain the key differences among EvoSuite, Randoop, and ChatGPT in terms of approach, generated test quality, and coverage.

6. **Tool Preference:** If you could only use one approach, which would you choose and why?

7. **Improvement Ideas:** Suggest features or improvements you'd like to see (e.g., better oracle generation, improved readability, handling of dependencies, etc.).

8. **Coverage Table:** Provide a small summary table comparing coverage metrics between all three methods.

Deliverables

1. Github Repository link consisting of Source code of generated tests (EvoSuite, Randoop, and ChatGPT).

The github directory structure must look like the **Maven project structure** shown below:

```
├──── pom.xml
├──── README.md
│
├──── src/
│     └──── Stack.java
│
└──── test/
   ├──── chatgpt/
   │     └──── ChatGPTTest.java
   ├──── randoop/
   │     ├──── RegressionTest0.java
   │     └──── ErrorTest0.java
   └──── evosuite/
      ├──── EvoSuiteTest.java
      └──── EvoSuiteRegressionTest.java
```

2. PDF report answering all questions above and a link to the public github repository.

The submission on e-learning must be Report pdf only.

Grading Criterion

Component	**Points**
Successful test generation with all three tools	25
Correct execution and coverage measurement	25
Quality and completeness of report	30
Thoughtfulness of comparison and insights	15
Presentation and clarity	5
Total	**100**