CSE 565: Software Verification and Validation

Structural-Based Testing Assignment

Purpose

This project aims to develop your skills in structural-based testing by analyzing code coverage and identifying data flow anomalies using appropriate tools and techniques.

Objectives

By completing this assignment, you will:

- Perform code coverage analysis using statement and decision coverage techniques.
- Create a comprehensive set of test cases to fulfill specified requirements.
- Detect and analyze data flow anomalies in source code.

Project Overview

This assignment is divided into two parts:

1. **Part 1: Code Coverage Analysis**
   You will work with a Java program (`VendingMachine.java`) and use a tool to measure statement and decision coverage. The goal is to identify untested areas and refine your test cases to improve coverage.
2. **Part 2: Data Flow Analysis**
   Using a second Java program (`StaticAnalysis.java`), you will perform static code analysis to detect and understand data flow anomalies such as uninitialized variables and potential data leaks.

Instructions

1. **Download the Materials**
   Retrieve the following files from the assignment page:
   - `VendingMachine.java`
   - `StaticAnalysis.java`

Part 1: Code Coverage Analysis

1. **Tool Selection**
   Research and select a code coverage analysis tool capable of measuring statement and decision coverage.

2. **Understand the Program Requirements**
   The `VendingMachine.java` program should:
   - Accept an integer input (money provided by the user).
   - Allow the user to choose one of three products:
     - **Candy** (20 cents)
     - **Coke** (25 cents)
     - **Coffee** (45 cents)
   - Return the selected product and any remaining change.
   - Display the amount required if insufficient funds are provided, along with other products that can be purchased with the available amount.
3. **Develop Test Cases**
   - Write a comprehensive set of test cases based on the program requirements.
   - Execute the program with your test cases and analyze the resulting code coverage.
   - Iteratively refine your test cases to achieve at least 100% statement coverage and 90% decision coverage.
4. **Deliverables**
   - Provide a detailed report including your test cases, code coverage metrics, and observations.
   - Highlight any changes made to your test cases to improve coverage.

Part 2: Data Flow Analysis

1. **Tool Selection**
   Research and choose a static code analysis tool capable of detecting data flow anomalies.
2. **Understand the Program Inputs**
   The `StaticAnalysis.java` program accepts the following inputs:
   - **Weight of the package** (integer)
   - **Length of the package** (integer)
   - **Type of product** (string)
3. **Analyze the Program**
   - Use the selected tool to analyze `StaticAnalysis.java`.
   - Identify and document the two data flow anomalies embedded in the program.
   - Evaluate how well the tool identifies these anomalies, and comment on its performance.
4. **Assess the Tool**
   Evaluate the tool based on:
   - **Features and Functionalities**: Describe the capabilities and functionalities provided by the tool.
   - **Anomaly Coverage**: Assess the types of data flow anomalies the tool can identify.
   - **Ease of Use**: Comment on the user experience, including the ease of setup, execution, and interpretation of results.