Assignment 5

Xtreme Team
Dennis Grajo
Mike McColm
Stephen Obandinma
Hojun Lee

White Box Test # 1 – Withdraw

1. **Code Section Being Tested**

```
# Attempts to withdraw an amount from an account
def withdraw(self, fromAccNum, amount):
    accountActive = self.accountsHash[fromAccNum][2]
1   if accountActive == 1:
        newBal = self.accountsHash[fromAccNum][0] - amount
2       if newBal >= 0:
            self.accountsHash[fromAccNum] = [newBal, self.accountsHash[fromAccNum][1].rstrip(), 1]
        else:
            print("Failed Constraint: insufficient funds")

    else:
        print("Failed Constraint: Account was deleted")
```

2. **Test Case Analysis**
   For the withdraw method, the team has decided to use **decision coverage**. Since there are two if statements (marked above with 1 and 2), there will be 4 possible test cases. (True-True, True-False, False-True, False-False). These cases will cover all possible decisions in this method.
3. **Test Inputs**

Test	Decision 1 (accountActive == 1)	Decision 2 (newBal > 0)	TSF statement
T1	1: true	1: true	WDR 1234567 1000 0000000 ***
T2	1: false	1: true	WDR 2468246 1000 0000000 ***
T3	2: true	2: false	WDR 1234567 999999999 0000000 ***
T4	2: false	2: false	WDR 2468246 999999999 0000000 ***

4. **Test Results**

The following test results were observed from the testing inputs above.

Test	Expected Result	Actual Result	Success (Y/N)	Explanation
T1	Successful transaction, 1000 withdrawn from account 1234567.	Same as expected	Y	No changes necessary
T2	Unsuccessful transaction, "Failed Constraint: Account was deleted"	ERROR	N	Small change in hash lookup to avoid crashing
T2	Unsuccessful transaction, "Failed Constraint: Account was deleted"	Same as expected	Y	No changes necessary
T3	Unsuccessful transaction, "Failed Constraint: insufficient funds"	Same as expected	Y	No changes necessary
T4	Unsuccessful transaction, "Failed Constraint: Account was deleted"	Same as expected	Y	No changes necessary

*Note: The testing was done manually, where the TSF statement was manually entered and the results were compared to the expected results by inspection.

White Box Test # 2 – Create

1. **Code Section Being Tested**

```
# Creates an account
def create(self, accNum, accName):
    1 if accNum not in self.accountsHash:
    2     self.accountsHash[accNum] = [0, accName.rstrip(), 1]
```

2. **Test Case Analysis**
   For the create method, the team has decided to use statement coverage. Since there is only one statement plus the if statement, only two mutants need to be created as test inputs.
3. **Test Inputs**

Test	Statement	Valid (Y/N)	TSF statement
T1a	1	Y	NEW 1234321 000 0000000 Bob #where accNum 1234321 doesn't exist)
T1b	1	N	NEW 1234567 000 0000000 Bob #where accNum 1234567 exists)
T2	2	Y	NEW 1234321 000 0000000 Bob #where accNum 1234321 doesn't exist)

4. **Test Results**

The following test results were observed from the testing inputs above.

Test	Expected Result	Actual Result	Success (Y/N)	Explanation
T1a	Valid account created and added to VAF/MAF	Account created (as expected)	Y	No changes necessary
T1b	Invalid account still created and added to VAF/MAF	Invalid account created (as expected)	Y	No changes necessary
T2	No change to accountsHash or VAF/MAF	No change (as expected)	Y	No changes necessary

*Note: The testing was done manually, where the TSF statement was manually entered and the results were compared to the expected results by inspection.

Work Distribution

Member	Estimated Hours	Assignment Aspects
Dennis Grajo	5	Creating test cases and test inputs for the desired methods to test
Mike McColm	5	Small edits to created test cases and performing/analyzing the actual tests and results.
Stephen Obadinma	4	Analyzing code for errors and ensuring the correctness of test cases.
Hojun Lee	4	Debugging of created test cases and verification to its requirements