Assignment 5
Due: Friday 11:59pm on Nov 22, 2024

Group Members:

Member #1:

First Name: Eric
Last Name: Morse
Student ID: 1141504

Member #2:

First Name: Or
Last Name: Brener
Student ID: 1140102

Member #3:

First Name: Danindu
Last Name: Marasinghe
Student ID: 1093791

1. Chapter 8.1, Question 4 i) $p = a \wedge (\neg b \vee c)$, provide solutions for (a) through (f) (13 Marks)
   a. List the clauses that go with predicate p (1 mark)

a, b, c

   b. Compute (and simplify) the conditions under which each clause determines predicate p (2 marks)

$p_a = p_{a=true} \oplus p_{a=false}$

$p_a = T \wedge (\neg b \vee c) \oplus F \wedge (\neg b \vee c)$

$p_a = (\neg b \vee c) \oplus F$

$p_a = ((\neg b \vee c) \vee F) \wedge (\neg(\neg b \vee c) \vee \neg F)$

$p_a = (\neg b \vee c) \wedge \neg F$

$p_a = (\neg b \vee c) \wedge T$

$p_a = (\neg b \vee c)$

Therefore: a **determines** p **when** $b = F \vee c = T$**.**

$p_b = p_{b=true} \oplus p_{b=false}$

$p_b = a \wedge (\neg T \vee c) \oplus a \wedge (\neg F \vee c)$

$p_b = a \wedge (F \vee c) \oplus a \wedge (T \vee c)$

$p_b = a \wedge c \oplus a \wedge T$

$p_b = a \wedge c \oplus a$

$p_b = a \oplus a \wedge c$

$p_b = a \oplus c \wedge a$

$p_b = a \wedge \neg c$

Therefore b **determines** p **when** $a = T \wedge c = F$**.**

$p_c = p_{c=true} \oplus p_{c=false}$

$p_c = a \wedge (\neg b \vee T) \oplus a \wedge (\neg b \vee F)$

$p_c = a \wedge (T) \oplus (a \wedge \neg b)$

$p_c = a \oplus (a \wedge \neg b)$

$p_c = a \oplus (\neg b \wedge a)$

$p_c = a \wedge (\neg\neg b)$

$p_c = a \wedge b$

Therefore c determines p when $a = T \wedge b = T$.

c. Write the complete truth table for each clause (4 marks)

Label your rows starting from 1. Use the format in the example underneath the definition of Combinatorial Coverage in Section 8.1.1. That is, row 1 should be all clauses true. You should include columns for the conditions under which each clause determines the predicate, and also a column for the value of the predicate itself.

Row	a	b	c	a **determines** p	b **determines** p	c **determines** p	$a \wedge (\neg b \vee c)$
1	T	T	T	T	F	T	T
2	T	T	F	F	T	T	F
3	T	F	T	T	F	F	T
4	T	F	F	T	T	F	T
5	F	T	T	T	F	F	F
6	F	T	F	F	F	F	F
7	F	F	T	T	F	F	F
8	F	F	F	T	F	F	F

d. List all pairs of rows from your table that satisfy General Active Clause Coverage (GACC) with respect to each clause. (2 marks)

When a is the active clause: Any of rows (1, 3, 4) paired with any of rows (5, 7, 8)
When b is the active clause: rows {2, 4}
When c is the active clause: rows {1, 2}

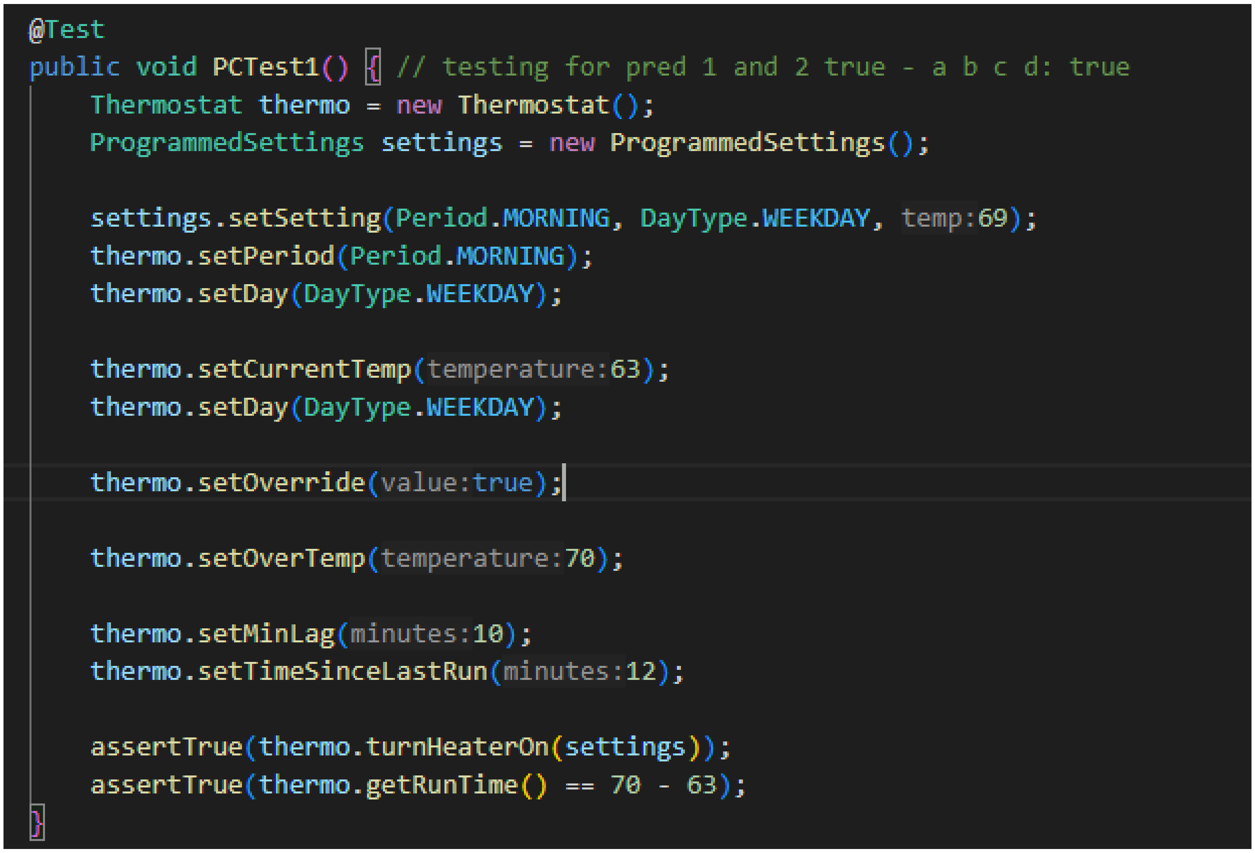
e. List all pairs of rows from your table that satisfy Correlated Active Clause Coverage (CACC) with respect to each clause. (2 marks)

When a is the active clause: Any of rows (1, 3, 4) paired with any of rows (5, 7, 8)
When b is the active clause: rows {2, 4}
When c is the active clause: rows {1, 2}

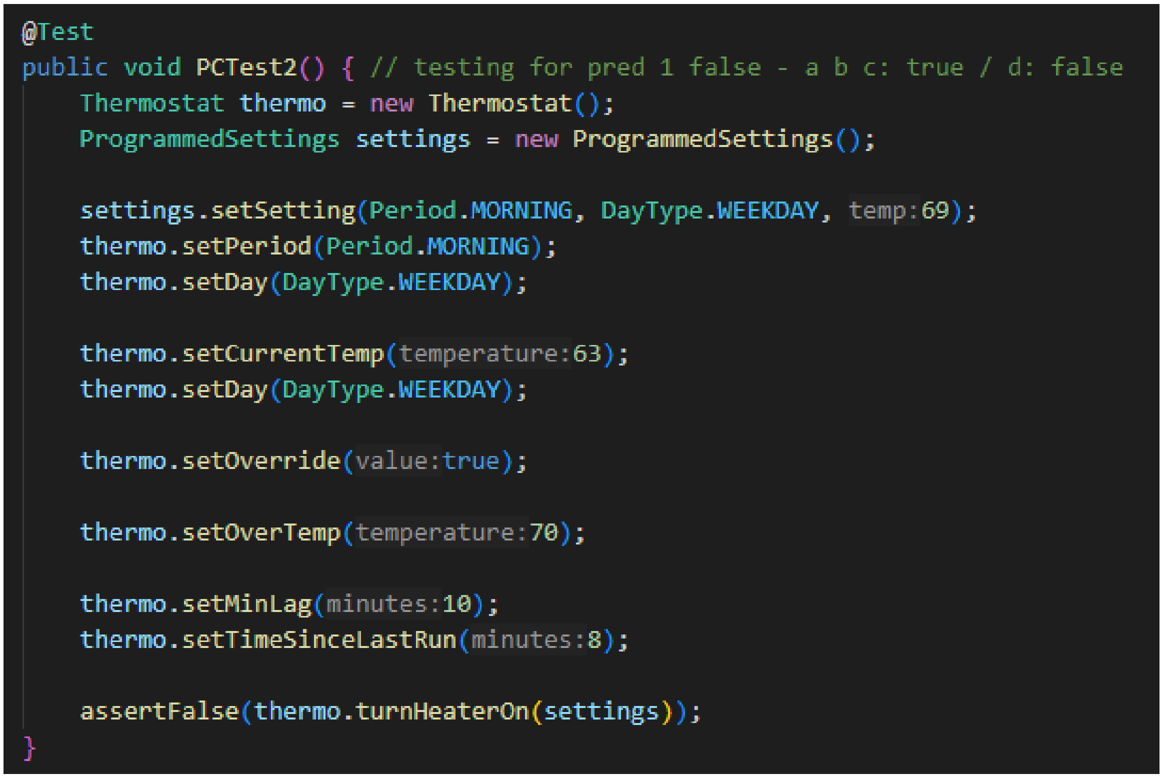
f. (f) List all pairs of rows from your table that satisfy Restricted Active Clause Coverage (RACC) with respect to each clause. (2 marks)

When a is the active clause: rows {1, 5}, {3, 7}, {4, 8}
When b is the active clause: rows {2, 4}
When c is the active clause: rows {1, 2}

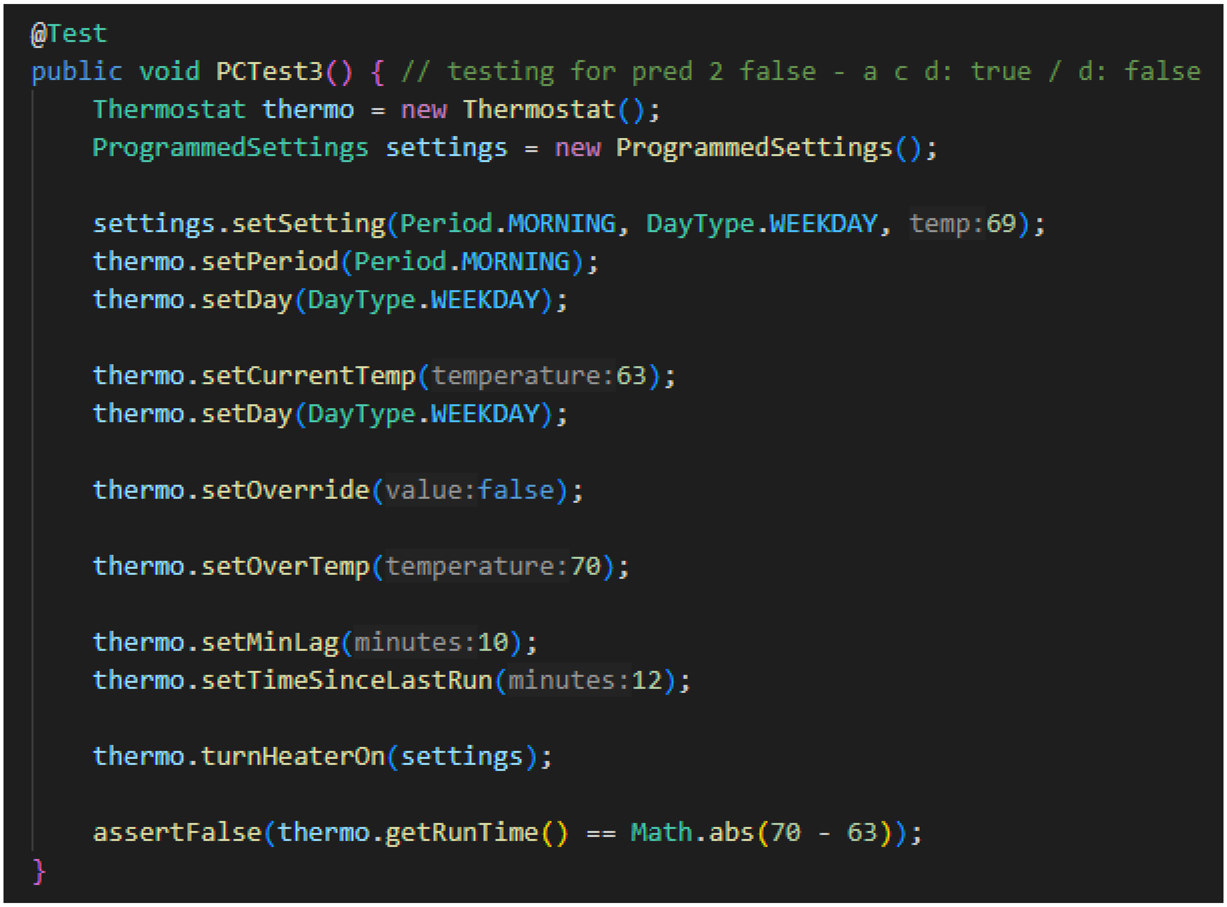
2. Chapter 8.3, question 1. Paste a screenshot of your program and test results below (2 marks).



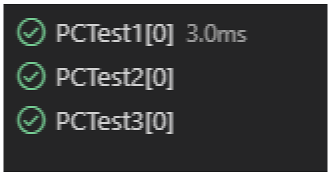
```
@Test
public void PCTest1() { // testing for pred 1 and 2 true - a b c d: true
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:69);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);

    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertTrue(thermo.turnHeaterOn(settings));
    assertTrue(thermo.getRunTime() == 70 - 63);
}
```



```java
@Test
public void PCTest2() { // testing for pred 1 false - a b c: true / d: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:69);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);

    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:8);

    assertFalse(thermo.turnHeaterOn(settings));
}
```

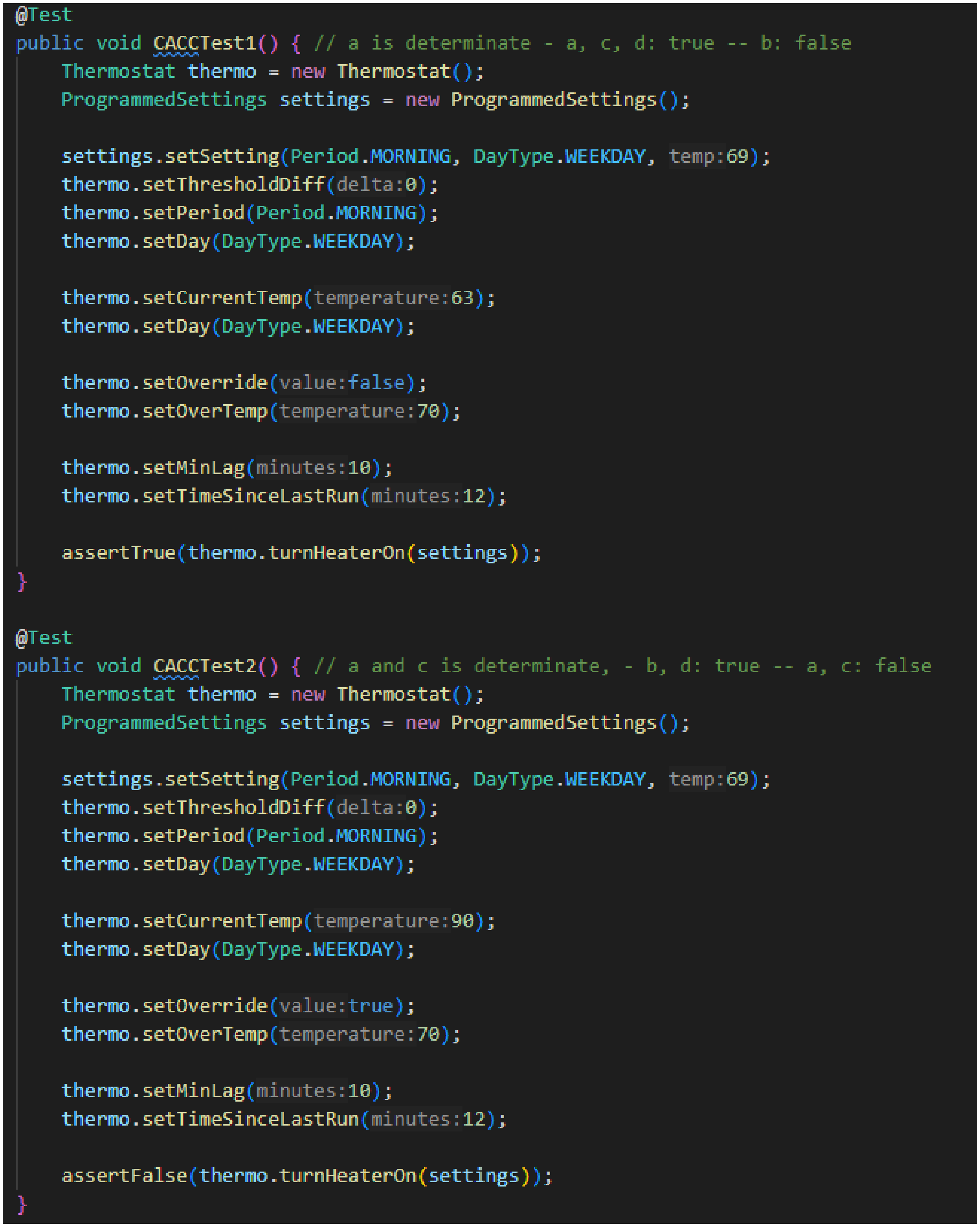


```java
@Test
public void PCTest3() { // testing for pred 2 false - a c d: true / d: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:69);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:false);

    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    thermo.turnHeaterOn(settings);

    assertFalse(thermo.getRunTime() == Math.abs(70 - 63));
}
```

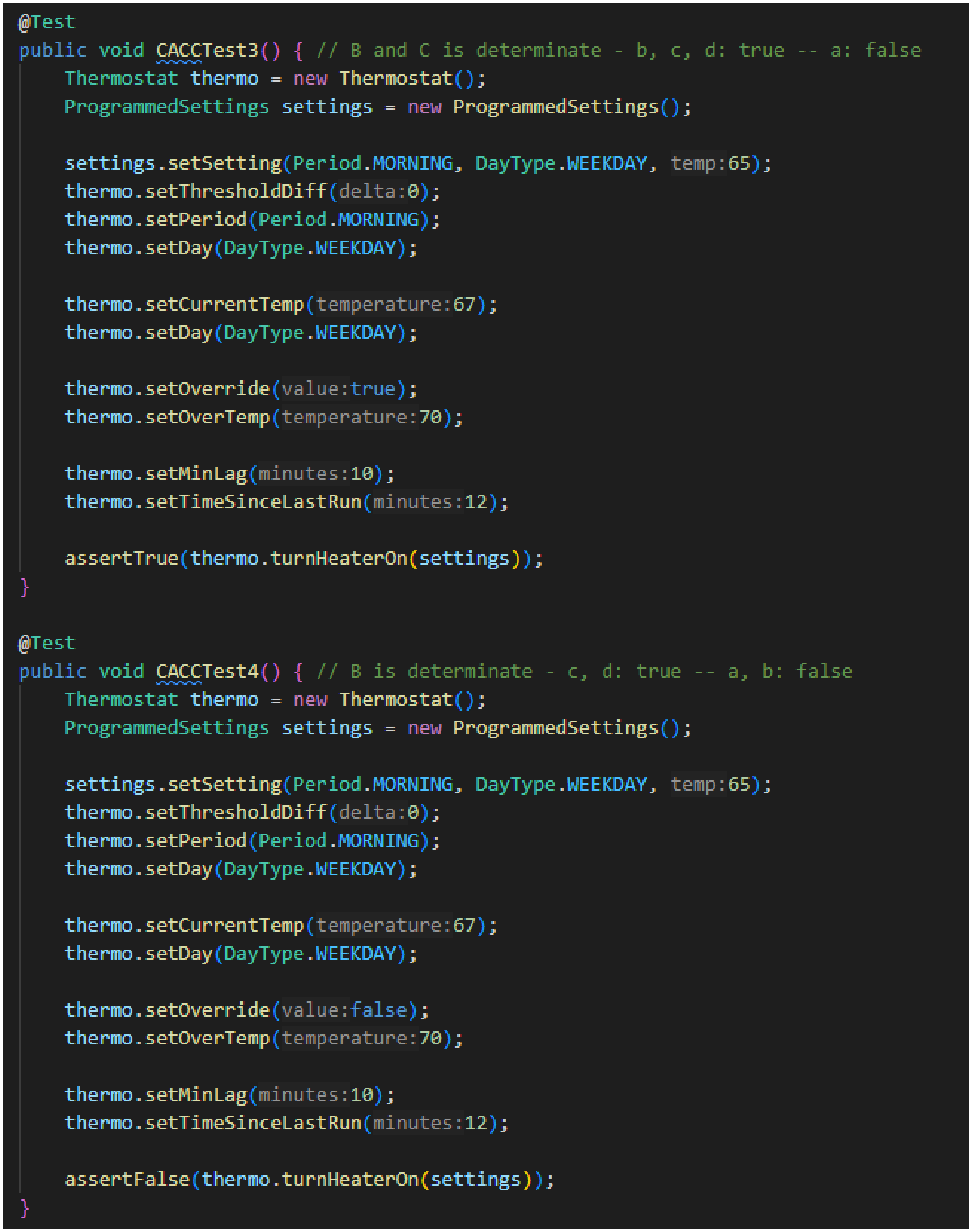


```
✓ PCTest1[0]  3.0ms
✓ PCTest2[0]
✓ PCTest3[0]
```

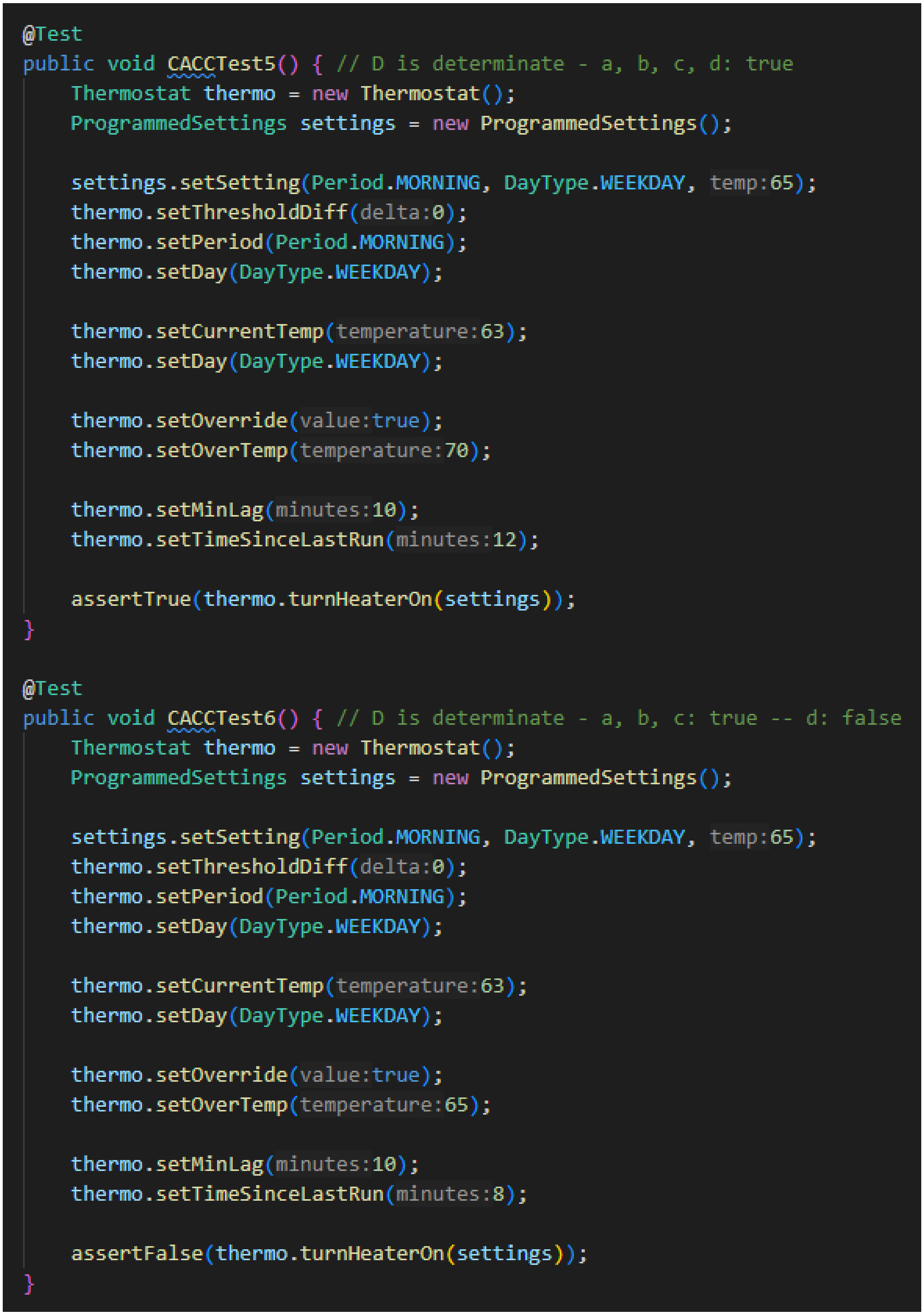
3. Chapter 8.3, question 3. Paste a screenshot of your program and test results below (3 marks).



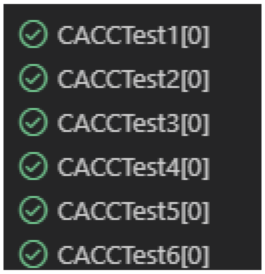
```
@Test
public void CACCTest1() { // a is determinate - a, c, d: true -- b: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:69);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:false);
    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertTrue(thermo.turnHeaterOn(settings));
}

@Test
public void CACCTest2() { // a and c is determinate, - b, d: true -- a, c: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:69);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:90);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);
    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertFalse(thermo.turnHeaterOn(settings));
}
```



```java
@Test
public void CACCTest3() { // B and C is determinate - b, c, d: true -- a: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:65);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:67);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);
    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertTrue(thermo.turnHeaterOn(settings));
}

@Test
public void CACCTest4() { // B is determinate - c, d: true -- a, b: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:65);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:67);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:false);
    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertFalse(thermo.turnHeaterOn(settings));
}
```



```java
@Test
public void CACCTest5() { // D is determinate - a, b, c, d: true
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:65);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);
    thermo.setOverTemp(temperature:70);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:12);

    assertTrue(thermo.turnHeaterOn(settings));
}

@Test
public void CACCTest6() { // D is determinate - a, b, c: true -- d: false
    Thermostat thermo = new Thermostat();
    ProgrammedSettings settings = new ProgrammedSettings();

    settings.setSetting(Period.MORNING, DayType.WEEKDAY, temp:65);
    thermo.setThresholdDiff(delta:0);
    thermo.setPeriod(Period.MORNING);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setCurrentTemp(temperature:63);
    thermo.setDay(DayType.WEEKDAY);

    thermo.setOverride(value:true);
    thermo.setOverTemp(temperature:65);

    thermo.setMinLag(minutes:10);
    thermo.setTimeSinceLastRun(minutes:8);

    assertFalse(thermo.turnHeaterOn(settings));
}
```

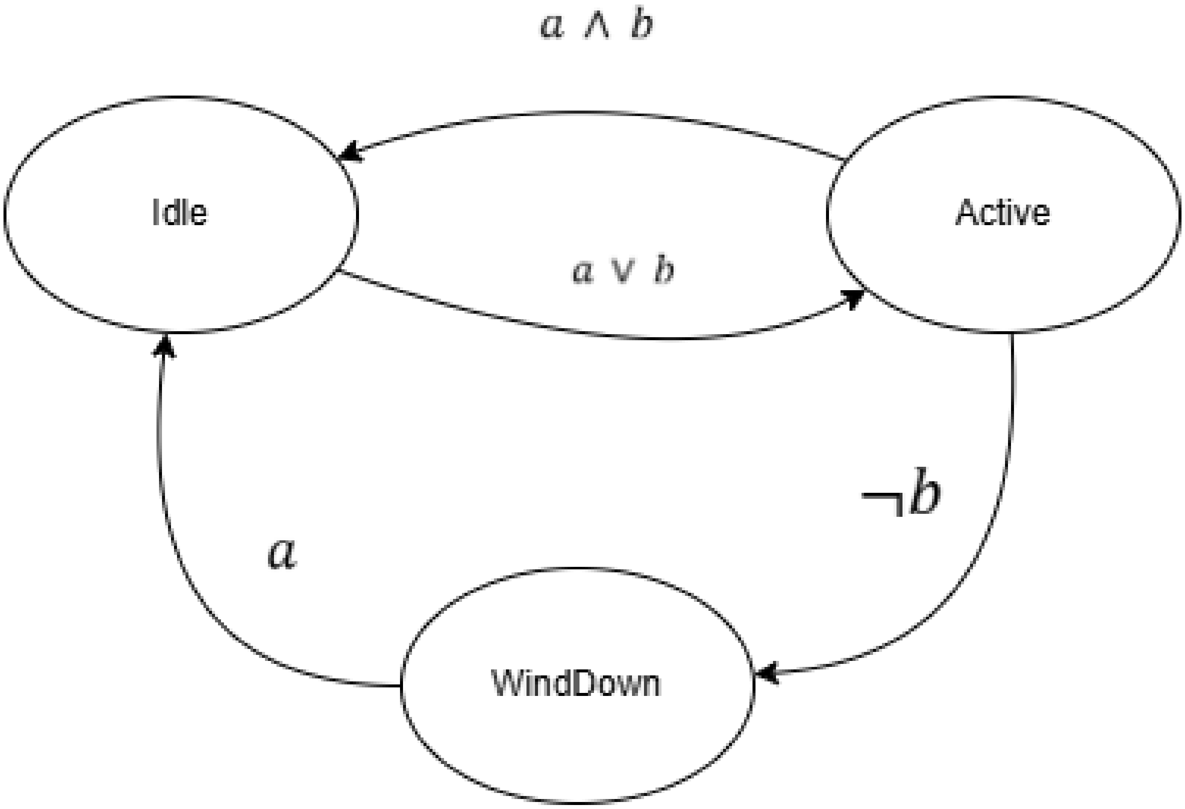


```
✓ CACCTest1[0]
✓ CACCTest2[0]
✓ CACCTest3[0]
✓ CACCTest4[0]
✓ CACCTest5[0]
✓ CACCTest6[0]
```

4. Chapter 8.5, question 5: Consider the following deterministic finite state machine (8 marks):

Current State	Condition	Next State
Idle	$a \vee b$	Active
Active	$a \wedge b$	Idle
Active	$\neg b$	WindDown
WindDown	a	Idle

a. Draw the finite state machine (2 marks)



b. This machine does not specify which conditions cause a state to transition back to itself. However, these conditions can be derived from the existing conditions. Derive the conditions under which each state will transition back to itself. (3 marks)

Current State	Condition	Transitions back to
Active	a	Active
WindDown	$a \wedge \neg b$	WindDown
Idle	a	Idle

c. Find CACC tests for each transition from the Active state (including the transition from Active to Active). (3 marks)

Active → Idle

a	b	Active → Idle $(a \wedge b) \vee (\neg b \wedge a)$
T	T	T
T	F	T
F	T	F
F	F	F

Tests where a is the major clause: {T, T}, {F, F}
Tests where b is the major clause: {T, T}, {F, F}

Thus, to satisfy CACC for Active → Idle, we can use the tests: {a = T, b = T}, {a = F, b = F}

Active → WindDown

b	Active → WindDown $\neg b$
T	F
F	T

There are no tests where a is the major clause.

Thus, to satisfy CACC for Active → WindDown, we can use the tests: {a = T, b = T}, {a = T, b = F}

Active → Active

a	Active → Active a
T	T
F	F

There are no tests where b is the major clause

Thus, to satisfy CACC for Active → WindDown, we can use the tests: {a = T, b = T}, {a = F, b = T}