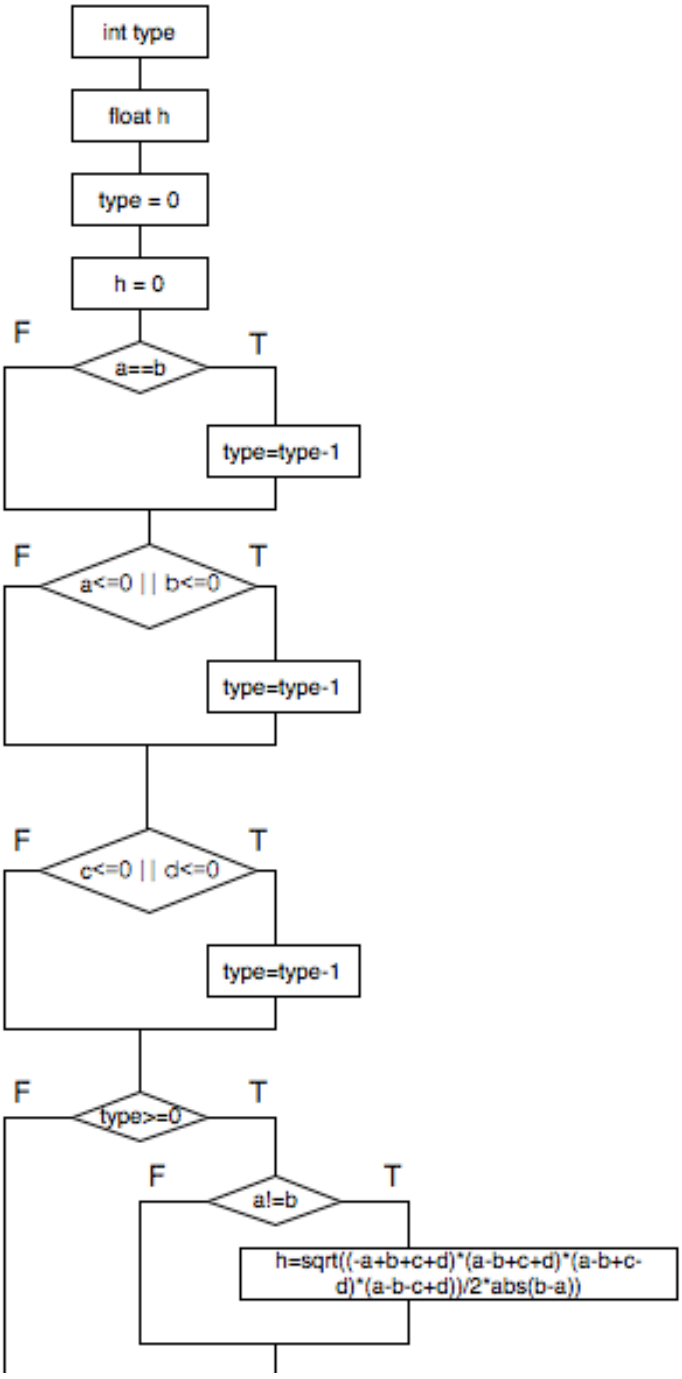
CS 589 Homework 2

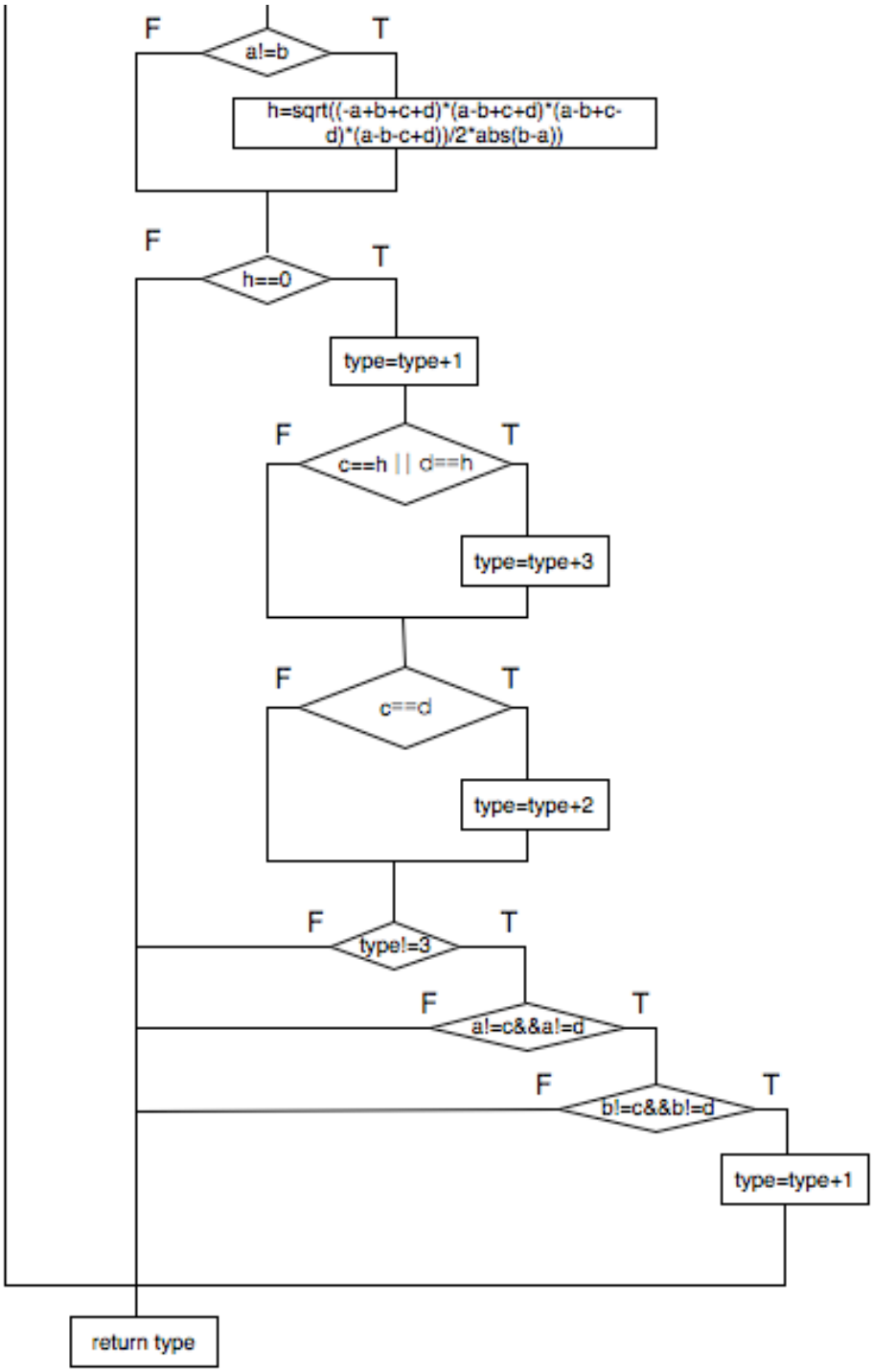
Xiaoyang Lu

A20315603

1.

a. For function trapezoid_type() derive a set of test cases that covers branch testing (all branches are executed). Show that your test cases execute all branches, i.e., for each branch show which test executes this branch.





Tests:

Test #1: a=6, b=6, c=8, d=3

Test #2: a=4, b=0, c=3, d=5

Test #3: a=4 b=3, c=0, d=5

Test #4: a=3, b=6, c=4, d=5

Test #5: a=3, b=11, c=5, d=5

Test #6: a=15, b=40, c=15, d=20

Test #7: a=40, b=15, c=15, d=20

Test #8: a=8, b=3, c=2, d=3

b. Derive additional test cases that cover multiple-condition testing. Show that your test cases execute all combinations of simple conditions for all complex predicates, i.e., for each combination of simple conditions indicate which test "executes" this combination. Notice that there are 5 conditional statements with complex predicates.

There are multiple-condition tests in 6, 8, 15, 20, 21.

6:

a <= 0 || b <= 0	
T	T
T	F
F	T
F	F

Tests:

Test #1: a=0, b=0, c=1,d=2

Test #2: a=0, b=1, c=2,d=3

Test #3: a=1, b=0, c=2,d=3

Test #4: a=3, b=4, c=6,d=5

8:

c <= 0 || d <= 0	
T	T
T	F
F	T
F	F

Tests:

Test #1: a=1, b=2, c=0,d=0

Test #2: a=1, b=2, c=0,d=3

Test #3: a=1, b=2, c=3,d=0

Test #4: a=3, b=4, c=6,d=5

15:

c == h || d == h

T	T
T	F
F	T
F	F

Tests:

Non-executable test for the first one

Test #1: a=3, b=6, c=4,d=5

Test #2: a=3, b=6, c=5,d=4

Test #3: a=15, b=40 c=15,d=20

20:

a != c && a != d	
T	T
T	F
F	T
F	F

Tests:

Test #1: a=3, b=6, c=5,d=4

Test #2: a=15, b=40, c=20,d=15

Test #3: a=15, b=40, c=15,d=20

Non-executable test for the last one

21:

b != c && b != d	
T	T
T	F
F	T
F	F

Tests:

Test #1: a=3, b=6, c=5,d=4

Test #2: a=40, b=15, c=20,d=15

Test #3: a=40, b=15, c=15,d=20

Non-executable test for the last one

2. Data-flow (definition-use) testing.
For function trapezoid_type() design a set of test cases that covers data-flow testing:
(1) identify all data flows (definition-use pairs), and then
(2) derive a set of test cases that "cover" all data flows.

(1) type

definition	use
2	5
2	7
5	7
2	9
5	9
7	9
2	10
5	10
7	10
9	10
2	14
14	16
14	18
14	19
16	19
18	19
14	22
16	22
2	23
5	23
7	23
9	23
14	23
16	23
18	23

h

definition	use
12	13
12	15

Test #1:

a = 5, b = 6, c = 7, d = 8

h: 12 → 13 12 → 15

type: 2 → 10 2 → 14 14 → 19 14 → 22 22 → 23

Test #2:

a = -2, b = -2, c = -4, d = 8

type: 2 → 5 5 → 7 7 → 9 9 → 10 9 → 23

Test #3:

a = 3, b = -4, c = 4, d = 5

type: 2 → 7 7 → 10 7 → 23

Test #4:

a = 2, b = 3, c = -3, d = 6

type: 2 → 9 9 → 23

Test #5:

a = 2, b = 2, c = -2, d = 4

type: 5 → 9

Test #6:

a = 6, b = 6, c = 7, d = 8

type: 5 → 10 5 → 23

Test #7:

a = 2, b = 9, c = 3, d = 4

type: 2 → 23

Test #8:

a = 6, b = 9, c = 4, d = 5

type: 14 → 16 16 → 22 16 → 19

Test #9:

a = 3, b = 5, c = 3, d = 3

type: 14 → 18 18 → 19 18 → 23

Test #10:

a = 4, b = 8, c = 3, d = 6

type: 14 → 22 14 → 23

Test #11:

a = 4, b = 7, c = 4, d = 5

type: 16 → 23

3. State-based testing
The Vending_Machine component (class) supports the following operations:
void create()
void coin ()
void insert_cups(int k)
void coffee()
void cream ()
void cancel()
A simplified EFSM model for the Vending_Machine component is shown below:
a. Design a set of test cases so each transition is "covered" in the EFSM diagram. For each test case show which transitions are "covered".
b. Design additional test cases that satisfy the transition-pairing testing criterion. Show that your test cases “execute” all transition-pairs, i.e., for each transitionpair indicate which test executes this transition-pair.

a.

Test #1:

Create(), insert_cups(3), coin(), coin(), cream(), coffee()

T1, T3, T4, T8, T15, T14

Test #2:

Create(), coin(), insert_cups(1), coin(), cream(), cream() coffee(), insert_cups(0),

T1, T2, T3, T4, T15 , T16, T7, T11

coin(), insert_cups(1)

T12, T9

Test #3:

Create(), insert_cups(1), coin(), cream(), coin(), cancel()

T1, T3, T4, T14, T17, T13

Test #4:

Create(), inset_cups(1), coin(), cream(), coffee()

T1, T3, T4, T5, T10

Test #5:

Create(), insert_cups(3), coin(), coffee()

T1, T3, T4, T5

Test #6:

Create(), insert_cups(2), coin(), cancel()

T1, T3, T4, T6

b.

Idel

In: T1, T2, T3, T5, T6, T9, T13, T14

Out: T2, T3, T4

24 pairs

(T1, T2) (T1, T3) (T1, T4) (T2, T2) (T2, T3) (T2, T4) (T3, T2) (T3, T3)

(T3, T4) (T5, T2) (T5, T3) (T5, T4) (T6, T2) (T6, T3) (T6, T4) (T9, T2)

(T9, T3) (T9, T4) (T13, T2) (T13, T3) (T13, T4) (T14, T2) (T14, T3) (T14, T4)

Coin Inserted

In: T4, T8, T16

Out: T5, T6, T7, T8, T15

15 pairs

(T4, T5) (T4, T6) (T4, T7) (T4, T8) (T4, T15)

(T8, T5) (T8, T6) (T8, T7) (T8, T8) (T8, T15)

(T16, T5) (T16, T6) (T16, T7) (T16, T8) (T16, T15)

Cream

In: T15, T17

Out: T10, T13, T14, T16, T17

10 pairs

(T15, T10) (T15, T13) (T15, T14) (T15, T16) (T15, T17)

(T17, T10) (T17, T13) (T17, T14) (T17, T16) (T17, T17)

No Cups

In: T7, T10, T11, T12

Out: T9, T11, T12

12 pairs

(T7, T9) (T7, T11) (T7, T12)

(T10, T9) (T10, T11) (T10, T12)

(T11, T9) (T11, T11) (T11, T12)

(T12, T9) (T12, T11) (T12, T12)

Test #7:

T1, T2, T2, T3, T3, T4, T8, T5

Test #8:

T1, T3, T4, T5, T2

Test #9:

T1, T3, T4, T7, T12, T12, T11, T11, T9, T4

Test #10:

T1, T3, T4, T8, T8, T15, T17, T17, T16, T15, T13, T3

Test #11:

T1, T3, T4, T8, T7, T9, T3, T4, T15, T17, T14, T3, T4, T15, T16, T5, T3

Test #12:

T1, T3, T4, T8, T6, T4, T15, T10, T11

Test #13:

T1, T3, T4, T15, T17, T14, T4, T15, T10, T12

Test #14:

T1, T3, T4, T5, T4, T15, T13, T4, T15, T16, T6, T4,T15, T17, T10, T9, T4, T15, T16, T8, T6, T3