White Box Tests for the Car Insurance Premium Program

Software Testing and Quality Assurance

Joe Timoney

1

Car Insurance Example

Specification: The basic cost of an insurance premium for drivers is €500, however, this premium can increase or decrease depending on three factors: their age, their gender and their marital status.
The input gender is given by the character 'M' for male and 'F' for female.
Drivers that are below the age of 25, male and single face an additional premium increase of €1500. If a driver outside of this bracket is married or female their premium reduces by €200, and if they are aged between 46 and 65 inclusive, their premium goes down by €100.
Drivers below the age of 16 and greater than the age of 65 cannot be insured and will return a value of zero for the premium. Program error checking to prevent an illegal entry for gender will also return a value of zero for the premium.

2

Car Insurance Program

- **Specification:**
- Program inputs:
  - age: INT_MIN…15, 16…24; 25…44;45…65; 66…INT_MAX
  - gender: 'M'; 'F'; invalid input
  - married: True; False;
- Program Outputs:
  - Premium: 0, 200, 300, 400, 500, 2000

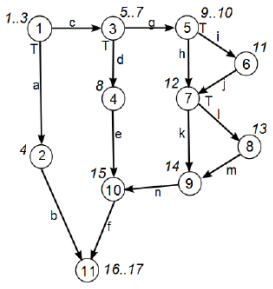
3

Program Code

```
(1)public int CarIns (int age, char gender, boolean married) {
(2) int premium;
(3) if ((age<16) || (age>65) || (gender!='M' && gender!='F')) {
(4) premium=0;
(5) } else {
(6)    premium=500;
(7)          If ((age<25) && (gender=='M') && (!married)) {
(8)                  premium += 1500;
(9)           } else {
(10)       If (married || gender=='F')
(11)         premium -= 200;
(12)           If ((age>=45) && (age<=65))
(13)                 premium -= 100;
(14)                  }
(15)               }
(16) return premium;
(17) }
```

4

CFG



5

Statement Testing

Test Data should cover the nodes 1..11

To further illustrate, the statements in the program can be identified as:

(2) int Premium;	Node 1
(4) Premium = 0;	Node 2
(6) Premium = 500;	Node 3
(8) Premium += 1500	Node 4
(11) Premium -= 200	Node 6
(13) Premium -= 100;	Node 8
(16) return Premium;	Node 11

Note: to execute these statements all the other nodes must be traversed too

6

Test Cases

Test Case	Node	Test
SC1	1	T4.1
SC2	2	T4.1
SC3	3	T4.2
SC4	4	T4.2
SC5	5	T4.3
SC6	6	T4.3
SC7	7	T4.3
SC8	8	T4.3
SC9	9	T4.3
SC10	10	T4.2
SC11	11	T4.1

7

Test Cases

1. Nodes (1) –(2) –(11)
2. Nodes 1–(3) –(4) –(10) –11
3. Nodes 1–3–(5) –(6) –(7) –(8) –(9) –10-11

All statements (nodes) in this method can be fully covered using 3 paths, shown with the non-duplicate nodes highlighted with a circle to assist in developing the test data:

8

Test Data

Test No.	Test Cases/Nodes Covered	Inputs			Expected Outputs
		age	*gender*	*Married*	*premium*
1	SC1, 2, 11	15	M	False	0
2	SC[1], 3, 4, 10, [11]	24	M	False	2000
3	SC[1, 3], 5, 6, 7, 8, 9, [10, 11]	46	F	False	200

9

Branch Testing – Test Cases

Case	Node	Test
BCa	a	T5.1
BCb	b	T5.1
BCc	c	T5.2
BCd	d	T5.2
BCe	e	T5.2
BCf	f	T5.2
BCg	g	T5.3
BCh	h	T5.3
BCi	i	T5.4
BCj	j	T5.4
BCk	k	T5.3
BCl	l	T5.4
BCm	m	T5.4
BCn	n	T5.3

10

Branch Testing Test Data

- All branches in this method can be fully covered using 4 paths, shown with the non-duplicate edges highlighted with a circle to assist in developing the test data:

1. Edges (a) –(b)
2. Edges (c) –(d) –(e) –(f)
3. Edges c–(g) –(h) –(k) –(n) –f
4. Edges c–g–(i) –(j) –(l) –(m) –n–f

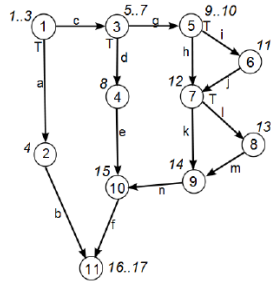
11

Test Data

Test No.	Test Cases/Branches Covered	Inputs			Expected Outputs
		Age	*gender*	*Married*	*Premium*
1	BCa, b	15	M	False	0
2	BCc, d, e, f	24	M	False	2000
3	BC[c], g, h, k, n, [f]	25	M	False	500
4	BC[c, g], I, j, l, m, [n, f]	45	F	True	200

12

Path Testing



1.(2+3.(4+5.(6+0).7. (8+0).9).10).11

13

Path Testing

If we wish to characterize the program using a Regular Expression we can write

- 1.(2+3.(4+5.(6+0).7. (8+0).9).10).11

Replace all values, including null, by 1 to compute the number of paths through the program. All iterations in the form (x)* become (x+0).This gives

•1.(2+3.(4+5.(6+0).7. (8+0).9).10).11
= 1.(1+1.(1+1.(1+1).1. (1+1).1).1).1
= 1.(1+1.(1+1.(2).1.(2).1).1).1
= 1.(1+(1+4).1)
= (1+ 5) =6

14

Path Coverage

The 6 possible paths through the program can be written

1) 1.2.11
2) 1.3.4. 10.11
3) 1.3.5.7.9.10.11
4) 1.3.5.6.7.9.10.11
5) 1.3.5. 7.8.9.10.11
6) 1.3.5.6.7.8.9.10.11

15

Test Cases

Each path is a separate test case

Case	Nodes	Test
P1	1.2.11	T10.1
P2	1.3.4.10.11	T10.2
P3	1.3.5.7.9.10.11	T10.3
P4	1.3.5.6.7.9.10.11	T10.4
P5	1.3.5.7.8.9.10.11	T10.5
P6	1.3.5.6.7.8.9.10.11	T10.6

16

Test Data

Test No.	Test Cases/Paths Covered	Inputs			Expected Outputs
		Age	*Gender*	*married*	*Premium*
1	P1	15	M	False	0
2	P2	24	M	False	2000
3	P3	25	M	False	500
4	P4	25	F	True	300
5	P5	45	M	False	400
6	P6	45	F	False	200

17

DU-Pairs

```
(1)public int CarIns (int age, char gender, boolean married) {
(2) int premium;
(3) if ((age<16) || (age>65) || (gender!='M' && gender!='F')) {
(4) premium=0;
(5) } else {
(6)    premium=500;
(7)          If ((age<25) && (gender=='M') && (!married)) {
(8)                  premium += 1500;
(9)           } else {
(10)      If (married || gender=='F')
(11)        premium -= 200;
(12)          If ((age>=45) && (age<=65))
(13)                premium -= 100;
(14)                  }
(15)              }
(16) return premium;
(17) }
```

18

Data Flow/DU Pairs

The principle in du-pair testing is to execute each path between the definition of the value in a variable and its subsequent use

Variables

```
1)age
2)gender
3)married
4)premium
```

19

DU Pairs

du-pairs for age

d-u pair	D	U	Test
DUP1	1	3	T11.1
DUP2	1	7	T11.2
DUP3	1	12	T11.3

du-pairs for gender

d-u pair	D	U	Test
DUP4	1	3	T11.1
DUP5	1	7	T11.2
DUP6	1	10	T11.3

20

DU Pairs

du-pairs for married

d-u pair	D	U	Test
DUP7	1	7	T11.2
DUP8	1	10	T11.3

21

DU Pairs

du-pairs for premium

d-u pair	D	U	Test
DUP9	6	8	T11.2
DUP10	6	11	T11.4
DUP11	6	13	T11.5
DUP12	11	13	T11.4
DUP13	4	16	T11.1
DUP14	6	16	T11.3
DUP15	8	16	T11.2
DUP16	11	16	T11.6
DUP17	13	16	T11.5

22

Test Data

Test No.	Test Cases/Pairs Covered	Inputs			Expected Outputs
		age	*gender*	*Married*	*premium*
1	DUP1, 4, 13	15	M	False	0
2	DUP2, 5, 7, 9, 15	24	M	False	2000
3	DUP3, 6, 8, 14	25	M	False	500
4	DUP10, 12, 17	45	F	True	200
5	DUP11	45	M	False	400
6	DUP16	25	F	False	300

23