Car Insurance Program – Black Box Testing

Program (2)

```
public int CarIns (int age, char gender, boolean married) {
int premium;
if ((age<16) || (age>65) || (gender!=’M’ && gender!=’F’)) {
premium=0;
} else {
   premium=500;
          If ((age<25) && (gender==’M’) && (!married)) {
                    premium += 1500;
             } else {
       If (married || gender==’F’)
          premium -= 200;
            If ((age>=45) && (age<=65))
                   premium -= 100;
                     }
                 }
return premium;
}
```

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 45 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.

Black Box Testing

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 45 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.

Car Insurance Example

- **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

Partitions

- **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

Input Partitions

Parameter	Test Case	Partition Range
Age	1*	INT_MIN…15
	2	16…24
	3	25…44
	4	45…65
	5*	66... INT_MAX
gender	6	M
	7	F
	8*	Invalid input
married	9	True
	10	False

Note: * indicates an "error" case. INT_MIN is the minimum possible integer value and INT_MAX is the maximum possible integer value.

Test Data

Test No.	Test Cases Covered	Inputs			Expected Outputs
		age	*gender*	*married*	*premium*
1	4, 7, 9, 12	50	F	True	200
2	3, 7, 10, 13	30	F	False	300
3	4, 6, 10, 14	50	M	False	400
4	3, 6, 10, 15	30	M	False	500
5	2, 6, 10, 16	20	M	False	2000
6*	1	5	M	False	0
7*	5	70	M	False	0
8*	8	50	G	False	0

Output Partitions

Parameter	Test Case	Partition Range
premium	11	0
	12	200
	13	300
	14	400
	15	500
	16	2000

Boundary Value Analysis

Having identified the Equivalence Partitions, it is straightforward to identify the Boundary Values at the lower and upper end of each Partition.

The input boundary values are given on the next slide

Test Cases

Each partition becomes a test case

Parameter	Test Case	Partition Boundary Value
age	1*	INT_MIN
	2*	15
	3	16
	4	24
	5	25
	6	44
	7	45
	8	65
	9*	66
	10*	INT_MAX
gender	11	M
	12	F
	13*	invalid input
married	14	True
	15	False

Note: * denotes error cases

Output Boundary Values

Parameter	Test Case	Boundary
Premium	16	0
	17	200
	18	300
	19	400
	20	500
	21	2000

Test Data

Test No.	Test Cases Covered	Inputs			Expected Outputs
		age	*gender*	*married*	*premium*
1	7, 12, 14, 17	45	F	True	200
2	5, 12, 14, 18	25	F	True	300
3	8, 11, 15, 19	65	M	False	400
4	6, 11, 15, 20	44	M	False	500
5	4, 11, 15, 21	24	M	False	2000
6	3, 11, 15, 21	16	M	False	2000
7*	1	INT_MIN	M	False	0
8*	2	15	M	False	0
9*	9	66	M	False	0
10*	10	INT_MAX	M	False	0
11*	13	16	G	False	0

Boundary Value Analysis

- **Note:** by definition, Boundary Value Analysis covers all the Equivalence Partition test cases.

Truth Tables

- To identify a minimum subset of possible combinations that will test all the different behaviours of the program, a Truth Table is created.
- The inputs ("Causes") and outputs ("Effects") are specified as Boolean expressions (using predicate logic); these expressions specify the conditions required for a particular variable.
- Test Cases are then constructed, one for each rule in the Truth Table.

Causes

The number of causes should be minimized to reduce the size of the Truth Table – in particular, where a parameter has a range of values that provide a particular response, this can be expressed as a single cause. The causes for this program, taken from the specification, can be expressed as follows:

- age<16
- 16<=age<=24
- 25<=age<45
- 45<=age<=65
- age>65
- gender='M'
- gender='F'
- Married==true

Note: it is not necessary to include the cause 'age>65' as this must be true if all the other possible value ranges for age are false. However, for clarity it is included here.

Effects

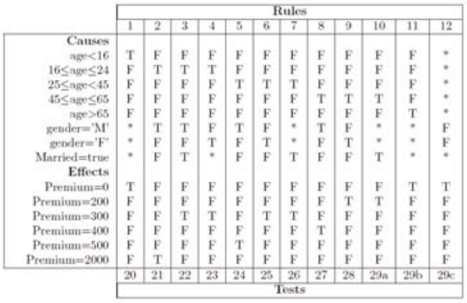
- Premium=0
- Premium=200
- Premium=300
- Premium=400
- Premium=500
- Premium=2000

Truth Table

- To generate the Truth Table, each Cause is listed in a separate row,
- Then a different column is used to identify each combination of Causes that creates a different Effect.
- Each column is referred to as a Rule in the Truth Table – each Rule is a different test case.

Truth Table

a '*' Indicates don't care



	Rules											
	1	2	3	4	5	6	7	8	9	10	11	12
Causes												
age<16	T	F	F	F	F	F	F	F	F	F	F	*
16≤age≤24	F	T	T	T	F	F	F	F	F	F	F	*
25≤age<45	F	F	F	F	T	T	T	F	F	F	F	*
45≤age≤65	F	F	F	F	F	F	F	T	T	T	F	*
age>65	F	F	F	F	F	F	F	F	F	F	T	*
gender='M'	*	T	T	F	T	F	*	T	F	*	*	F
gender='F'	*	F	F	T	F	T	*	F	T	*	*	F
Married=true	*	F	T	*	F	F	T	F	F	T	*	*
Effects												
Premium=0	T	F	F	F	F	F	F	F	F	F	T	T
Premium=200	F	F	F	F	F	F	F	F	T	T	F	F
Premium=300	F	F	T	T	F	T	T	F	F	F	F	F
Premium=400	F	F	F	F	F	F	F	T	F	F	F	F
Premium=500	F	F	F	F	T	F	F	F	F	F	F	F
Premium=2000	F	T	F	F	F	F	F	F	F	F	F	F
	20	21	22	23	24	25	26	27	28	29a	29b	29c
	Tests											

Test Cases

- Each Rule is a Test Case, and needs to be tested in a separate test.
- The test data is derived by picking values that satisfy the Causes and Effects for a Rule.

Test Data

Test No.	Test Cases/Rule Covered	Inputs			Expected Outputs
		age	*gender*	*married*	*Premium*
1	1	15	M	False	0
2	2	20	M	False	2000
3	3	20	M	True	300
4	4	20	F	True	300
5	5	36	M	False	500
6	6	36	F	False	300
7	7	36	M	True	300
8	8	50	M	False	400
9	9	50	F	False	200
10	10	50	M	True	200
11	11	70	M	False	0
12	12	36	'G'	True	0