Software Testing

UCL Buddy

Assignment 2 | Section 2

BSE173017 – Mohammad Abdur-Rahman
BSE173004 – Abdul Moiz
BSE173025 – Abdul Basit
BSE173041 – Huraira Ali

Table of Contents

Case Study 2
Introduction 2
Brief Description 2
Identified Functions 2
Function I 2
Function II 2
Function III 2
Black-Box Testing 3
Worst Case BVA 3
Function I 3
Function II 4
Function III 4
Strong Robust Equivalence Classes 5
Function I 5
Function II 5
Function III 5
Comparison between strong robust equivalence class and robust worst case BVA 6
Function 1 6
Function II 6
Function III 6

Case Study

Introduction

UEFA Champions league is football competition where there are many teams that play against each other to win the champions league title. In this competition top 2 to 4 teams from all Europeans football leagues take part and the winner team is called the best team in Europe and is selected for the Club World Cup to get the Club world Cup title and also the winner team is give almost 85m Dollars as reward price followed by 65m Dollars for runner ups.

Brief Description

In this application the player performance is tracked i.e. his goals, assists and matches are counted by the application. Goals assist ratio (G/A) is calculated by the sum of number of goals a player scores and the number of assists he has given in a match where goals are valued twice as assists. Goal range in the champions' league is 0 to 15(including 0 and 15), and number of assists could be from 0 to max 10 in the competition. Also, G/A can't be calculated if the number of matches is 0. G/A can be calculated by adding Goals and assists and dividing the number with the total matches played

Player rating is calculated by using the G/A, if the rating is low, we will write bad season, average season for average rating and great season for high rating

Tracking system is also used to track the progress of the goal keeper and calculate that how much clean sheet a keeper keeps in a number of matches. A clean sheet is kept by a keeper if he doesn't let the opposition team score against him. Also, there can be only one clean sheet per match so the number of matches may be equal to or higher than the clean sheets a keeper kept. If the numbers of clean sheets are higher than matches then the output should be invalid.

Identified Functions

Function I

float playerGA (float matches,float goal, float assist)

In this function G/A will be calculated. It will take goals assists and matches played as inputs to calculate G/A. If matches are 0 G/A is invalid and goals value double in comparison to assists in GA.

Function II

void rankCalculation (float KDA, float BD) {//mentioned in the code present on GitHub}

This will tell about the performance of the player using the G/A

Function III

float cleansheetratio (float matches, float cs)

It will calculate the cleansheet ratio of the goalkeeper. Number of matches must be less than or equal to the number of clean sheets kept by a keeper. Also, competition consists of 7 matches so clean sheets should not exceed 7

Black-Box Testing

Worst Case BVA

Function I

float playerGA (float matches, float goal, float assist)

Total test cases= 5^3 =125
Half test cases implemented = 125/2=62.5~62
Input Values:

Matches: min = 0, min+ = 1, nom = 4, max- = 6, max = 7
Goal: min = 0, min+ = 1, nom = 8, max- = 14, max = 15
Assist: min = 0, min+ = 1, nom = 5, max- = 9, max = 10
Expected Output calculated: G/A=((2*goals) +assist)/matches

Case	Matches	Goals	Assist	Expected Output
1	0	0	0	N/A
2	0	0	1	N/A
3	0	0	5	N/A
4	0	0	9	N/A
5	0	0	10	N/A
6	0	1	0	N/A
7	0	1	1	N/A
8	0	1	5	N/A
9	0	1	9	N/A
10	0	1	10	N/A
11	0	8	0	N/A
12	0	8	1	N/A
13	0	8	5	N/A
14	0	8	9	N/A
15	0	8	10	N/A
16	0	14	0	N/A
17	0	14	1	N/A
18	0	14	5	N/A
19	0	14	9	N/A
20	0	14	10	N/A
21	0	15	0	N/A
22	0	15	1	N/A
23	0	15	5	N/A
24	0	15	9	N/A
25	0	15	10	N/A
26	1	0	0	0
27	1	0	1	1
28	1	0	5	5
29	1	0	9	9
30	1	0	10	10
31	1	1	0	1

Case	Matches	Goals	Assist	Expected Output
32	1	1	1	3
33	1	1	5	7
34	1	1	9	11
35	1	1	10	12
36	1	8	0	16
37	1	8	1	17
38	1	8	5	21
39	1	8	9	25
40	1	8	10	26
41	1	14	0	28
42	1	14	1	29
43	1	14	5	33
44	1	14	9	37
45	1	14	10	38
46	1	15	0	30
47	1	15	1	31
48	1	15	5	35
49	1	15	9	39
50	1	15	10	40
51	4	0	0	0
52	4	0	1	0.25
53	4	0	5	1.25
54	4	0	9	2.25
55	4	0	10	2.5
56	4	1	0	0.5
57	4	1	1	0.75
58	4	1	5	1.75
59	4	1	9	2.75
60	4	1	10	3
61	4	8	0	2
62	4	8	1	2.25

Function II

float GAratio (int GA)

Total test cases= 5^1 =5
All cases implemented = 5
Input Values:
G/A: min = 0, min+ = 1, nom = 20, max- = 39, max = 40

Case	G/A	Expected Output
1	0	Invalid
2	1	Bad Season
3	20	Great season
4	39	Great season
5	40	Great Season

Function III

void Cleansheetratio (float Matches, float CS)

Total test cases = 5^2 =25
Half test cases implemented = 12
Input Values:
Matches: min = 0, min+ = 1, nom = 4, max- = 6, max = 7
CS: min = 0, min+ = 1, nom = 4, max- = 6, max = 7

Case	Matches	CS	Expected Output
1	0	0	0
2	0	1	N/A
3	0	4	N/A
4	0	6	N/A
5	0	7	N/A
6	1	0	0
7	1	1	1
8	1	4	N/A
9	1	6	N/A
10	1	7	N/A
11	4	0	0
12	4	1	0.25

Strong Robust Equivalence Classes

Function I

float playerGA (float matches, float goal, float assist)

Total test cases = 30
Half test cases implemented = 15
Test data = Enter 3 inputs.
Pre-condition = Matches 0 to 7, Goals 0 to 15 and Assist 0 to 10

Case	Matches	Goals	Assist	Expected Output
1	2	1	10	6
2	1	-1	1	N/A
3	-1	3	2	N/A
4	2	3	-1	N/A
5	-2	-2	3	N/A
6	-2	4	-2	N/A
7	3	-3	-1	N/A
8	-2	-3	-2	N/A
9	2	17	2	N/A
10	8	5	5	N/A
11	3	6	12	N/A
12	7	18	5	N/A
13	8	10	12	N/A
14	2	20	13	N/A
15	7	16	11	N/A

Function II

float GAratio (float GA)

Total test cases = 3
Test data = Enter value of GA
Pre-condition = GA 0 to 40

Case	G/A	Expected Output
1	0	Invalid
2	9	Bad season
3	40	Great season

Function III

void Cleansheetratio (float Matches, float CS)

Total test cases = 5
Test data = Enter 2 inputs values of type float.
Pre-condition = KDA 6 to 20 and (Baron and dragon ratio) BD 0 to 3

Case	Matches	CS	Expected Output
1	0	2	Invalid
2	3	1	0.33
3	3	4	Invalid
4	7	7	1
5	8	6	Invalid

Comparison between strong robust equivalence class and robust worst case BVA

Function 1

float playerGA (float matches, float goal, float assist)

In function 1 the strong robust equivalence class has **15 test cases** of the program while robust worst case BVA has **343 (7^3) test cases** of the program.

Function II

float GAratio (float GA)

In function 2 the strong robust equivalence class has **3 test cases** of the program while robust worst case BVA has **7 (7^1) test cases** of the program.

Function III

void Cleansheetratio (float Matches, float CS)

In function 3 the strong robust equivalence class has **5 test cases** of the program while robust worst case BVA has **25 (5^2) test cases** of the program.

As you can see there is a huge difference between them. The reason is that strong robust equivalence class identifies classes and identifies test cases within and beyond the boundary with the help of those classes, which reduces the test cases while in robust worst case BVA it has no classes which makes more test cases generated and it checks within the boundaries and also beyond the boundaries. Hence cause more test cases as compared to strong robust equivalence class.