Software Testing HW1

[github](https://github.com/oiu850714/software-test)

**Triangle Problem**

**Boundary Value Test**

寫好 testcase，尚未實作 function 之前:

<https://i.imgur.com/6B4fsXF.png>

<https://i.imgur.com/zinKUmF.png>

寫了 robust case 的實作後:

<https://i.imgur.com/F9UvBvT.png>

<https://i.imgur.com/Xy6G4eP.png>

把剩下的邏輯判斷都寫完後:

<https://i.imgur.com/p4k67KZ.png>

<https://i.imgur.com/SUm39sk.png>

再加一些 special value testing，這樣才有判斷是一般三角形(Scalene)的情況:

<https://i.imgur.com/qPnzgkR.png>

中途我發現因為還要寫其他三種 test methods 的 test cases，於是我決定要更改 test case 跟 test 的名稱:

<https://i.imgur.com/vhXMJiK.png>

<https://i.imgur.com/BBlLHJe.png>

**Equivalence Class Test**

<https://i.imgur.com/9MklkvX.png>

<https://i.imgur.com/CDKtPSo.png>

加上 Equivalence class test，目前的實作還是都正確

**Edge value Test**

我認為這個 Test 應該不用再額外寫? 因為他是 boundary 跟 equivalence 的 hybrid，上面都已經有寫 test cases 了

**Decision Table Test**

我照著 ch07 講義的第 7, 22 頁寫出以下測資；decision table 應該不用分 weak/strong normal/robust?

<https://i.imgur.com/ydHde2q.png>

<https://i.imgur.com/vqMydH4.png>

**NextDate Problem**

註: 因為 googletest 的 macro，遇到 function 直接吃 C++ 的 initailizer\_list 的情況會抽筋，所以只好把年月份包成一個比較漂亮的 class，而不是寫一個 struct Date{int year, month, day} 然後初始化給 initializer\_list 就好。

* 相關 issue: https://github.com/google/googletest/issues/1022

**Boundary Value Test**

單純定義一個 class Date 並且 overload operator==

<https://i.imgur.com/QBmengi.png>

提供一個空的實作:

<https://i.imgur.com/HVEWWbY.png>

先寫 normal/robust/random boundary value test:

<https://i.imgur.com/o08uTd6.png>

<https://i.imgur.com/2QK9EDN.png>

空實作的結果:

<https://i.imgur.com/xVjKXbx.png>

* 註: robust 的因為 input 就 invalid，所以 output 全部都 invalid，而我空的實作原本就是 return 一個 invalid object，所以 test 會過

**Equivalence Class Test**

直接抄課本比較快，這邊我的 equivalence classes 如下:

M1 = { month: month has 30 days}

M2 = { month: month has 31 days}

M3 = { month: month is February}

D1 = {day: 1 ≤ day ≤ 28}, D2 = {day: day = 29}

D3 = {day: day = 30}, D4 = {day: day = 31}

Y1 = {year: year = 2000 }

Y2 = {year: year is a non-century leap year},

Y3 = {year: year is a common year}

直接寫 strong normal，這裡 robust 我覺得沒有意義就不實作了，因為結果全部都會是 invalid input

test:

<https://i.imgur.com/Jqjkyk0.png>

結果

<https://i.imgur.com/8aUTemu.png>

寫 decision table 前先把實作寫好試試看:

https://i.imgur.com/xerPtZ7.png

https://i.imgur.com/QIvDpnW.png

於是目前的 test 就全過了

https://i.imgur.com/llUYndk.png

**Decision Table Test**

直接用講義第七章 p.28 的 third try

M1 = {month: month has 30 days}

M2 = {month: month has 31 days except December}

M3 = {month: month is December}

M4 = {month: month is February}

D1 = {day: 1 ≤ day ≤ 27}

D2 = {day: day = 28}

D3 = {day: day = 29}

D4 = {day: day = 30}

D5 = {day: day = 31}

Y1 = {year: year is a leap year}

Y2 = {year: year is a common year}

按照 third try 的 decision table 實際給值後的 code:

<https://i.imgur.com/Uflmv3M.png>

結果:

<https://i.imgur.com/0RgkW9i.png>

**Commission Problem**

我沒有像課本範例那樣，讓 user 一直輸入 locks, stocks, barrels 的值然後用 locks == -1 當作 EOF；我直接一次把這三個分別的數量輸入完。

一樣先提供空的實作:

<https://i.imgur.com/oQoerWd.png>

<https://i.imgur.com/Y3HUBmV.png>

**Boundary Value Test**

直接按照課本的 test cases 填: 這裡參考課本 p.91-92 頁，**用 output 的區間來區分 boundary**，分別是 commission value 為 1000 跟 1800 附近的值

<https://i.imgur.com/unrvqUJ.png>

<https://i.imgur.com/x6eQ0vu.png>

這裡主要是切 output，而 output 變數只有一個，所以 strong test 不會有額外測資；然後 robust test，也就是要讓 output 為 invalid，只需要讓任意 input 為 invalid 就行了:

<https://i.imgur.com/84I3OTJ.png>

目前空的實作的測試結果:

<https://i.imgur.com/f0Sqhd8.png>

實作了確認 input 區間的 code 之後，robust test 就過了:

<https://i.imgur.com/kdD4TGG.png>

<https://i.imgur.com/2ZN1KNS.png>

**Random Test**

加一些 random tests:

<https://i.imgur.com/QCVvKdb.png>

<https://i.imgur.com/7oT5klM.png>

**Equivalence Class Test**

這裡如果用把 input domain 的 legal 跟 illegal 區間分開成不同的 classes 的方法來找 test case 的話，那基本上會得到跟 boundary value 差不多的測資(講義第六章 p.27 頁那樣，或者課本 p.111 說的)

所以我用課本介紹的另一種分法: 切 output 區間:

S1 = {<locks, stocks, barrels>}: sales < 1000

S2 = {<locks, stocks, barrels>}: 1000 <= sales <= 1800

S3 = {<locks, stocks, barrels>}: sales > 1800

這樣分的話 testcase 也只有三個:

<https://i.imgur.com/8NtpvUi.png>

<https://i.imgur.com/cZXdf70.png>

寫 Decision Table test 之前先把實作完成:

<https://i.imgur.com/867thLX.png>

目前的 Test 都通過了:

<https://i.imgur.com/snF2ls4.png>

**Decision Table Test**

按照課本說法，這個範例幾乎沒有什麼 decision，所以用這個方法生的測資也很難測到什麼東西，而且硬要寫的話，其 test cases 會跟 equivalence class 得到的 test cases 一樣

<https://i.imgur.com/a8Xj5e9.png>

<https://i.imgur.com/5dWiOCM.png>

<https://i.imgur.com/sKqVSES.png>