

程式設計理念和規格書

第一題

程式設計理念：

Quadrilateral 為主程式部分, 由 check 對輸入值檢查是否符合範圍, 再由 partition 判斷其四邊形種類。再利用不同的 test cases, 對程式進行測試。

系統規格：

A. boundary testing

測試目的: 測試在規格範圍中(valid), 系統的可靠度

測試規格:

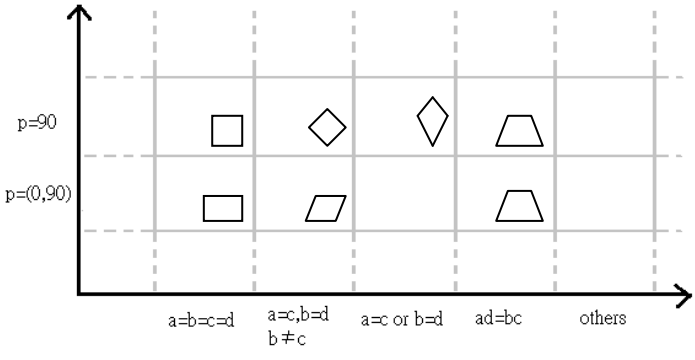
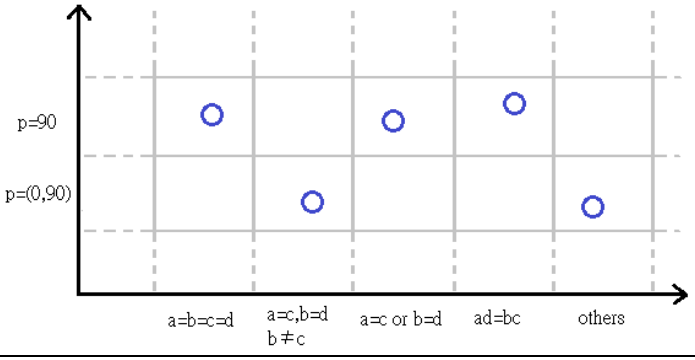
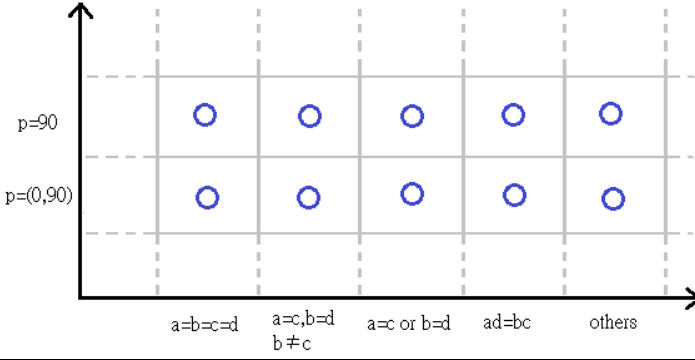
Boundary	1. 規格中的 a, b, c, d 為對角線, 其下限為 1, 上限為 100 2. 規格中的 p 為夾角, 範圍限制為(0,90] 3. 測試時分別取 a, b, c, d, p 為邊界值, 略大於下限值, 及略小於上限值來做測試
Robustness	1. a, b, c, d, p 的範圍限制與 boundary 相同 2. 考慮 invalid 的部分, 因此, 除了 boundary 的測試項目, 分別在 a, b, c, d, p 取略小於下限值, 及略大於上限值來做測試

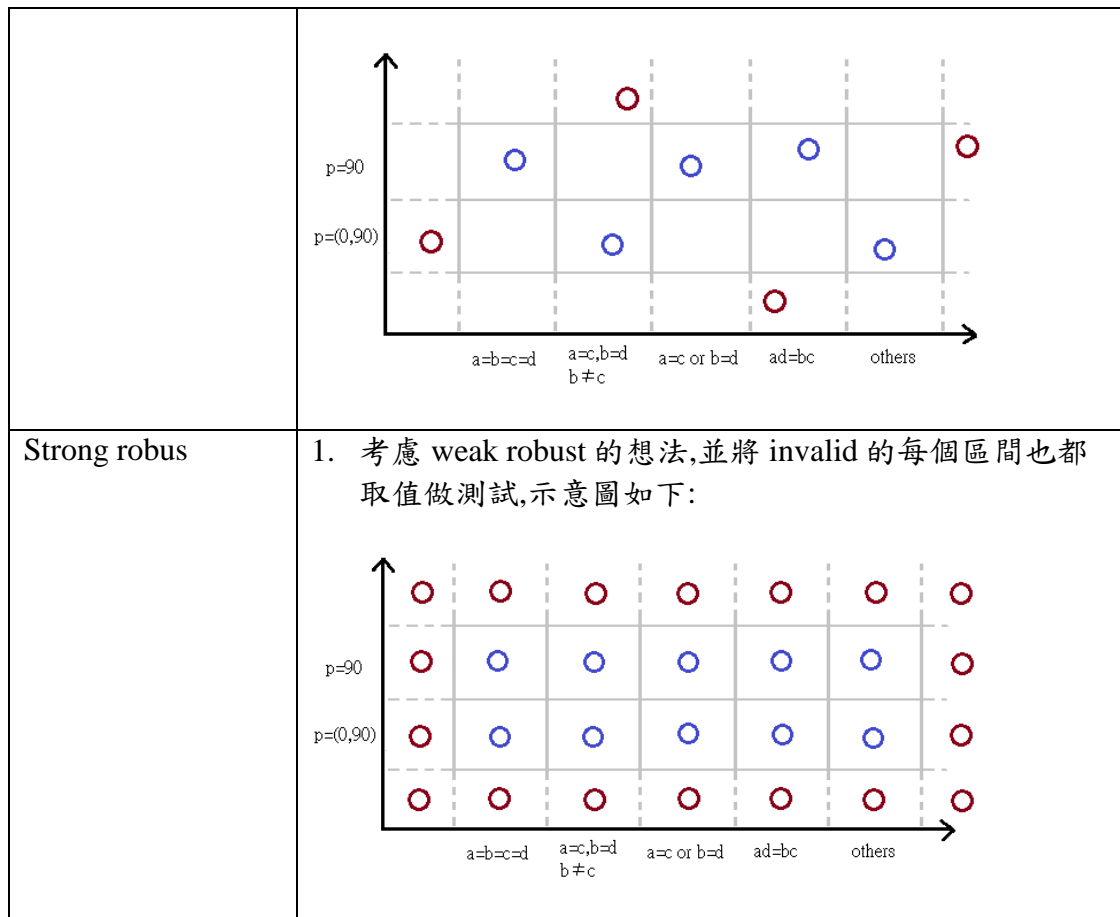
B. equivalence classes

測試目的: 測試在規格範圍中(valid), 以及範圍之外(invalid), 系統的可靠度

測試規格:

Weak normal	1. 從輸入的值, 我們可以得到 7 種不同的四邊形類型, 用下面所列出的條件去做分類: (1) $a=b=c=d$ (2) $a=c, b=d, b \neq c$ (3) $a=c$ or $b=d$ (4) $ad=bc$ (5) $p=90$ 2. 以 a, b, c, d 的條件判斷為 x 軸, p 的值為 y 軸, 示意圖如下:
-------------	---

	 <p>3. 從上圖, 取 weak normal, 每個區間都要有取到對應的值, 因此至少取 5 組可以滿足測試, 示意圖如下:</p> 
Strong normal	<p>1. 考慮方式與 weak normal 相同</p> <p>2. 在 valid 的每個區間, 都要取值, 示意圖如下:</p> 
Weak robust	<p>1. 除了 weak normal 所考慮的部分, 也要在 invalid 的區間取值, 因此在 a, b, c, d, p 各值小於下限的部分, 以及大於上限的部分, 也都取值來做測試, 示意圖如下:</p>



C. decision classes

測試目的:利用條件式將結果分類,由條件式的不同組合來對程式結果做測試

測試規格:由四邊形的分類規則,可以找出其特性:

Square	$a=b=c=d, p=90$
Rhombus	$a=c$ and $b=d, p=90$
Rectangle	$a=b=c=d, p=(0,90)$
harrier-shaped	$a=c$ or $b=d, p=90$
parallelogram	$a=c$ and $b=d, p=(0,90)$
trapezium	$ad=bc$
quadrilateral	else

我們可以從上面的規則,利用 5 個條件式,去判斷所能產生的圖形

Conditions:

C1: $a=c$

C2: $b=d$

C3: $a=b$












































































C4: $a*d=b*c$

C5: $P=90$

不過由於某些條件式無法同時成立,所以會產生 A8:impossible 的情形

C1:a=c	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F
C2:b=d	T	T	T	T	T	-	F	F	F	F	F	T	T	T	T	F
C3:a=b	T	T	T	F	F	F	T	T	F	F	-	-	-	T	F	-
C4:a*d=b*c	T	T	F	T	T	F	F	F	T	F	T	F	T	F	F	F
C5:P=90	T	F	-	T	F	-	T	F	-	-	-	T	-	F	F	-
Rule Count	1	1	2	1	1	4	1	1	2	2	4	2	4	1	1	4
A1: square	X															
A2: rhombus		X														
A3: rectangle				X												
A4: harrier-shaped							X					X				
A5: parallelogram					X											
A6: trapezium											X					
A7: quadrilateral								X		X					X	X
A8:impossible			X			X			X				X	X		

D. Coverage 與 Cyclomatic Complexity 計算方法:

statement and branch coverage	<div>code cover</div> <table><thead><tr><th>Name</th><th>Statement</th><th>Branch</th></tr></thead><tbody><tr><td> eec</td><td> 83.5 %</td><td> 79.2 %</td></tr><tr><td>▷  boundary_testing</td><td> 100.0 %</td><td> 50.0 %</td></tr><tr><td>▷  decision_table</td><td> 89.1 %</td><td> 50.0 %</td></tr><tr><td>▷  equivalence_SN</td><td> 85.9 %</td><td> 50.0 %</td></tr><tr><td>▷  equivalence_SR</td><td> 89.5 %</td><td> 85.7 %</td></tr><tr><td>▷  equivalence_WN</td><td> 74.0 %</td><td> 42.9 %</td></tr><tr><td>▷  equivalence_WR</td><td> 100.0 %</td><td> 100.0 %</td></tr><tr><td>▷  robustness_testing</td><td> 98.9 %</td><td> 50.0 %</td></tr></tbody></table>	Name	Statement	Branch	 eec	 83.5 %	 79.2 %	▷  boundary_testing	 100.0 %	 50.0 %	▷  decision_table	 89.1 %	 50.0 %	▷  equivalence_SN	 85.9 %	 50.0 %	▷  equivalence_SR	 89.5 %	 85.7 %	▷  equivalence_WN	 74.0 %	 42.9 %	▷  equivalence_WR	 100.0 %	 100.0 %	▷  robustness_testing	 98.9 %	 50.0 %
Name	Statement	Branch																										
 eec	 83.5 %	 79.2 %																										
▷  boundary_testing	 100.0 %	 50.0 %																										
▷  decision_table	 89.1 %	 50.0 %																										
▷  equivalence_SN	 85.9 %	 50.0 %																										
▷  equivalence_SR	 89.5 %	 85.7 %																										
▷  equivalence_WN	 74.0 %	 42.9 %																										
▷  equivalence_WR	 100.0 %	 100.0 %																										
▷  robustness_testing	 98.9 %	 50.0 %																										
Cyclomatic Complexity	<div>McCabe</div> <table><tbody><tr><td> quadrilateral</td><td></td></tr><tr><td>partition</td><td>16</td></tr><tr><td>check</td><td>11</td></tr><tr><td>main</td><td>1</td></tr></tbody></table> <p>使用 Metrics (plug-in of Eclipse) 可計算出程式的 Cyclomatic Complexity 在主程式當中 check(檢查輸入是否符合範圍條件)=11, partition(偵測輸入所能產生的四邊形)=16</p>	 quadrilateral		partition	16	check	11	main	1																			
 quadrilateral																												
partition	16																											
check	11																											
main	1																											