EJERCICIOS SOBRE PRUEBAS DE CAJA NEGRA

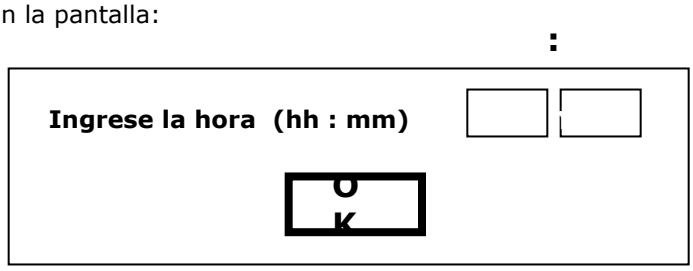
0.1. Un programa recibe como entrada un número entero y positivo de mínimo 2 cifras y de máximo 9 cifras y devuelve el número resultante de invertir sus cifras.

Si no se introduce un valor acorde a lo descrito (por ejemplo: flotantes y/o caracteres, valores fuera de rango, etc.), el módulo devolverá el valor **"error"**. Genere la tabla de clases de equivalencia y los casos de prueba (no olvide el análisis de valores límite).

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 1 } **1**	{ n < 1 } **2.1** { n > 1 } **2.2**
	B	Tipo de los parámetros	{ x ∈ N } **3**	{ x ∉ N } **4**
	C	Intervalo	{ x > 9 , x < 1000000000 } **5**	{ x < 10 } **6.1** { x > 999999999 } **6.2**

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	(456248)	842654	1 , 3 , 5	(10) (12) (999999999) (999999998)	1 21 999999999 899999999
Clases erróneas	( )	Error	2.1		
	(2148 , 215879 , 345872)	Error	2.2	(2148 , 215879)	Error
	(254.3689)	Error	4		
	('z')	Error	4		
	(8)	Error	6.1	(9)	Error
	(1000000001)	Error	6.2	(1000000000)	

0.2. Tenemos un pequeño módulo que lee una hora en formato de hora militar e indica si la hora es correcta. Se anexa a continuación la pantalla:



Si no se introduce un valor acorde a lo descrito (por ejemplo: flotantes y/o caracteres, valores fuera de rango, etc.), el módulo devolverá el valor **"error"**. Genere la tabla de clases de equivalencia y los casos de prueba (no olvide el análisis de valores límite).

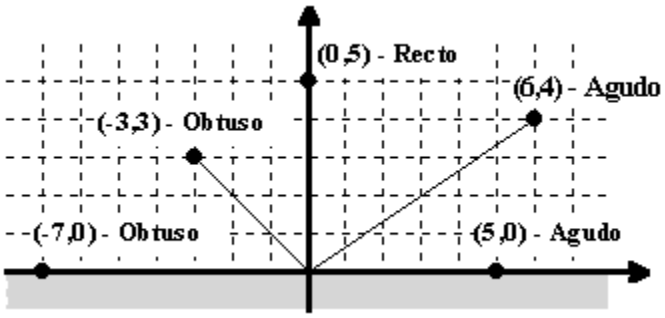
Solución:
n = número de parámetros
A = conjunto de valores permitidos: los números naturales más el cero. A = N **U** {0}.

Asume		Condición	Clases correctas	Clases erróneas
	A	N° de parámetros	{ n = 2 } **1**	{ n < 2 } **2**
	B	Tipo de los parámetros	{ hh ∈ A , mm ∈ A } **3**	{ hh ∉ A } **4.1** { mm ∉ A } **4.2**
A , B	C	HORA VALIDA	{ hh >= 0 , hh < 24 , mm >= 0 , mm < 60 } **5**	{ hh < 0 } **6.1** { hh > 23 } **6.2** { mm < 0 } **6.3** { mm > 59 } **6.4**

De la tabla de particiones obtenemos la siguiente batería de pruebas:

	Entradas	Salidas	Clases Cubiertas	Valores Límite	Salidas
Clases correctas	(16 , 43)	VALIDA	1 , 3 , 5	(0 , 0) (23 , 59)	VALIDA VALIDA
Clases erróneas	(,) ambos nulos	Error	2	(, 35) Uno nulo	Error
	(11.3 , 38)	Error	4.1		
	('h' , 38)	Error	4.1		
	(11 , 38.3)	Error	4.2		
	(11 , 'mm')	Error	4.2		
	(-3 , 45)	Error	6.1	(-1 , 45)	Error
	(26 , 12)	Error	6.2	(24 , 12)	Error
	(2 , -5)	Error	6.3	(2 , -1)	Error
	(2 , 63)	Error	6.4	(2 , 60)	Error

1. Se dispone de un módulo cuyo objetivo es clasificar como agudo, recto u obtuso el ángulo que forma una determinada línea con el eje horizontal. La línea se define como aquella que pasa por el origen de coordenadas y por un punto del semiplano superior. Las coordenadas de este punto son los valores que recibe el módulo como parámetro.



Para que el módulo pueda calcular un valor correcto (agudo, recto, obtuso) las entradas deben cumplir los siguientes requisitos:

- El módulo debe recibir como entrada dos valores enteros, que representan las coordenadas horizontal y vertical, respectivamente, del punto anteriormente descrito.
- El punto pertenece al semiplano superior o al eje horizontal.
- El punto a evaluar es diferente al origen de coordenadas.
- De no cumplirse los requisitos anteriores, el módulo devolverá el valor **"error"**.

Elaborar una batería de pruebas de caja negra para el módulo descrito.

SOLUCION. Las condiciones que separan entradas correctas de entradas erróneas son el número de parámetros, el tipo de los parámetros, que el punto pertenezca al semiplano superior y que no sea el origen. Parece lógico que se compruebe que no sea el punto origen antes de comprobar que pertenezca al semiplano superior, ya que la primera condición es más general que la segunda.

Ya que el módulo clasifica un ángulo, se supondrá que aquellas entradas que dan lugar a resultados distintos se tratan de forma diferenciada en el módulo, lo que da lugar a tres condiciones según la entrada pertenezca al semiplano derecho (ángulo agudo), a la ordenada vertical (ángulo recto) o al semiplano izquierdo (ángulo obtuso). Para estas condiciones, lo mínimo que se puede asumir es que el número y tipo de los parámetros es el adecuado. La tabla de particiones de equivalencia resultante sería la siguiente:

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 2 } **1**	{ n < 2 } **2.1** { n > 2 } **2.2**
	B	Tipo de los parámetros	{ x ∈ Z , y ∈ Z } **3**	{ x ∉ Z } **4.1** { y ∉ Z } **4.2**
	C	Punto origen	{ x ≠ 0 } **5.1** { y > 0 } **5.2**	{ x = 0 , y = 0} **6**
	D	Semiplano superior	{ y >= 0 } **7**	{ y < 0 } **8**
A , B	E	Angulo Agudo	{ **x > 0** , y >= 0} **9**	{ **x > 0** , y < 0} **10**
A , B	F	Angulo Recto	{ **x = 0** , y > 0} **11**	{ **x = 0** , y < 0} **12** { **x = 0** , y = 0} **12.a (6)**
A , B	G	Angulo Obtuso	{ **x < 0** , y >= 0} **13**	{ **x < 0** , y < 0} **14**

De la tabla de particiones se obtiene la siguiente batería de pruebas:

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	(2 , 2)	agudo	1 , 3 , 5.1 , 5.2 , 7, 9	(1 , 0)	Agudo
	(0 , 2)	recto	1 , 3 , 5.2 , 7 , 11	(0 , 1)	Recto
	(-2 , 2)	obtuso	1 , 3 , 5.1 , 5.2 , 7 , 13	(-1 , 0)	Obtuso
Clases erróneas	(2)	error	2.1		
	(2 , 2 , 2)	error	2.2		
	('a' , 2)	error	4.1		
	(2 , 'a')	error	4.2		
	(3.88 , 2)	error	4.1		
	(2 , 7.24)	error	4.2		
	(0 , 0)	error	6		
	(2 , -2)	error	8 , 10		
	(0 , -2)	error	8 , 12		
	(-2 , -2)	error	8 , 14		

02. Disponemos de un módulo **búsqueda**, cuya función es buscar un determinado valor entero dentro de un vector de enteros.

- En caso de encontrar el valor, el módulo devuelve la posición del vector donde se ha encontrado (al primer componente del vector le corresponde la posición 1, al segundo la posición 2, etc). Si existieran varias apariciones del valor buscado dentro del vector, el módulo devolverá la menor posición donde se encuentre.
- En caso de no encontrarse dentro del vector, el módulo devuelve -1.
- Si no se cumplen los requisitos anteriores, el módulo devolverá el valor **"error"**.

Los parámetros del módulo son, en primer lugar el elemento a buscar y a continuación el vector.

Ejemplos:

busqueda (5 , [1,9,5,5,3,5,9]) devolverá 3.
busqueda (-7 , [1,-7,3,2,-8,6,0]) devolverá 2.
busqueda (15 , [1,-7,3,2,-8,6,0]) devolverá -1.

Diseñar un conjunto de casos de prueba de caja negra.

SOLUCION. Algunos símbolos que vamos a utilizar:
n → Número de parámetros del módulo **Ve** → Vector de enteros **x** → elemento a buscar
Vc → Vector de caracteres

La tabla de particiones de equivalencia sería:

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 2 } **1**	{ n < 2 } **2.1** { n > 2 } **2.2**
	B	Tipo de los parámetros	{ x ∈ Z , Ve ∈ Z } **3**	{ x ∉ Z } **4.1** {Ve ∉ Z } **4.2**
A , B	C	Valor existe en el vector	{ (x ∈ Ve) } **5**	
	D	Valor no existe en el vector	{ (x ∉ Ve) } **6**	

Una posible batería (genérica) de casos de prueba consistente con la tabla anterior sería:

	Entradas	**Salidas**	**Clases Cubiertas**
Clases correctas	(2 , [1,-7,3,2,-8,6,10])	4	1 , 3 , 5
	(15 , [1,-7,3,2,-8,6,10])	-1	1 , 3 , 6
Clases erróneas	(2)	Error	2.1
	([1,-7,3,2,-8,6,10])	Error	2.1
	(15 , [1,-7,3,2,-8,6,10] , 8)	Error	2.2
	('a' , [1,-7,3,2,-8,6,10])	Error	4.1
	(9.54 , [1,-7,3,2,-8,6,10])	Error	4.1
	(15 , ['f' , 'z' , 'e' , 'k'])	Error	4.2

Valores Límite: Si las estructuras de datos tienen un límite preestablecido hay que diseñar casos que ejerciten estos límites.

Entradas	**Salidas**	**Clases Cubiertas**
(V[1] , Ve) (4 , [4,-7,3,2,-8,6,10])	1	1 , 3 , 5
(V[tamaño] , Ve) (10 , [4,-7,3,2,-8,6,10])	Tamaño 7	1 , 3 , 5

03. Elaborar una batería de pruebas para un módulo que recibe como entrada una cadena de caracteres y determina si puede ser una clave válida o no (por lo tanto devuelve un valor lógico, verdadero o falso). Una clave se considera válida si cumple los requisitos siguientes:

- Está formada por **más** de 4 y **menos** de 8 caracteres.
- Los caracteres permitidos son los alfabéticos a...z, A...Z, los dígitos 0...9 y los caracteres especiales '%' y '#'.
- Contiene al menos dos alfabéticos.
- Contiene al menos un carácter que es dígito.
- El primer y último carácter deben ser alfabéticos.
- No aparece en un diccionario de palabras prohibidas (user%10, us3r%aa, etc).

SOLUCION. En primer lugar, vamos a nombrar los siguientes elementos relacionados con la entrada para que la escritura de las condiciones sea más sencilla:

n : Número total de caracteres de la cadena.

c : la cadena de caracteres
c(i) : Carácter i-ésimo de la cadena.

K : Conjunto de caracteres que son alfabéticos → (a…z) + (A…Z)
E : Conjunto de caracteres especiales → '%' y '#'
AE: Conjunto del Alfabeto extendido → K + E
CD: Conjunto de caracteres que son dígitos → (0…9)
CP : Conjunto de caracteres permitidos → AE + CD

nCD : Número de caracteres de la cadena que son dígitos.
nK : Número de caracteres de la cadena que son alfabéticos.
D : Conjunto de cadenas de caracteres que pertenecen al diccionario de palabras prohibidas.

Está formada por **más** de 4 y **menos** de 8 caracteres.	n>4 , n<8
Caracteres permitidos son los alfabéticos a..z, A..Z, los dígitos 0..9 y los especiales '%' y '#'	c(i) ∈ **CP**
Contiene al menos dos letras.	nK >=2
Contiene al menos un carácter que es dígito.	nCD >= 1
El primer y último carácter deben ser alfabéticos.	c[1] ∈ K , c[n] ∈ K
No aparece en un diccionario de palabras prohibidas.	c ∉ D

Este problema es especial en el sentido de que el módulo sólo proporciona dos valores de la salida, verdadero o falso. Podemos suponer que el módulo establece como precondición que el número y tipo de los parámetros es el adecuado, ya que no existe un valor de la salida que permita indicar que ha detectado que un error respecto al número y tipo de los parámetros. Por otro lado, existen dos posibilidades a la hora de realizar el análisis:

- Suponer que no hay entradas "erróneas". En este caso existirían dos condiciones derivadas de las dos salidas distintas del módulo, clave válida y clave no válida.
- Suponer que las entradas "correctas" son aquellas que coinciden con claves válidas y las entradas "erróneas" son aquellas que coinciden con claves no válidas. En este caso sólo existirá una clase que distingue entre entradas "correctas" y "erróneas".

Eligiendo esta última posibilidad, la tabla de particiones de equivalencia sería la siguiente:

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	Validez de la clave	{ n>4 , n<8 , c(i) ∈ **CP** , nK >=2 , nCD >= 1 , c[1] ∈ K , c[n] ∈ K , c ∉ D } **1**	{ n <= 4 } **2.1** { n >= 8 } **2.2** { c(i) ∉ **CP** } **2.3** { nK < 2 } **2.4** { nCD = 0 } **2.5** { c[1] ∉ K } **2.6** { c[n] ∉ K } **2.7** { c ∈ D } **2.8**

La batería de pruebas que se obtiene es:

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	"ab12%Ec"	VERDADERO	1	"ab12c" "ab1234c"	VERDADERO
Clases erróneas	"a1b"	FALSO	2.1	"a1#b"	FALSO
	"a1234567b"	FALSO	2.2	"G12345%b"	FALSO
	"ab*1*c"	FALSO	2.3	"ab*12c"	FALSO
	"12345"	FALSO	2.4	"1u2345"	FALSO
	"abcdef"	FALSO	2.5		
	"1abcde"	FALSO	2.6		

	"abcde1"	FALSO	2.7		
	"us3r%aa"	FALSO	2.8		

04. Se desea probar un módulo que calcula el máximo común divisor de dos valores enteros mayores que cero que recibe como entrada. Se sabe que el módulo utiliza un algoritmo que requiere realizar comparaciones de igualdad, mayor y menor entre los parámetros. Si el módulo recibe un número de parámetros distinto de dos, los tipos de los parámetros no son los adecuados o alguno de ellos es negativo o cero, el resultado devuelto por el módulo es el valor -1.

Crear la tabla de particiones de equivalencia y la batería de pruebas que se deriva de ella, añadiendo los casos de prueba adicionales obtenidos por el análisis de valores límite.

SOLUCION. La tabla de particiones de equivalencia constará de tres condiciones asociadas a la naturaleza del problema, y otras tres derivadas del control de las entradas cuyos parámetros son iguales entre sí, el primero es mayor y el primero es menor se tratan de manera diferenciada en el módulo. Estas tres últimas condiciones deben asumir que el número de parámetros es dos y que su tipo de datos es entero, ya que si no la formulación de las condiciones carecería de sentido. Sin embargo, la condición de que sean mayores que cero no tiene por qué asumirse.

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 2 } **1**	{ n < 2 } **2.1** { n > 2 } **2.2**
	B	Tipo de los parámetros	{ a ∈ N , b ∈ N } **3**	{ a ∉ N } **4.1** / { b ∉ N } **4.2** { a ∈ Caracteres } **4.3** { b ∈ Caracteres } **4.4**
	C	Parámetros mayores que cero	{ a > 0 , b > 0 } **5**	{ a <= 0 } **6.1** {b <= 0 } **6.2**
A , B	D	Parámetros iguales	{ **a = b** , a > 0 } **7**	{ **a = b** , a <= 0 } **8**
A , B	E	Primer parámetro mayor	{ **a > b** , b > 0 } **9**	{ **a > b** , b <= 0 , a > 0 } **10.1** { **a > b** , b <= 0 , a <= 0 } **10.2**
A , B	F	Segundo parámetro mayor	{ **a < b** , a > 0 } **11**	{ **a < b** , a <= 0 , b > 0 } **12.1** { **a < b** , a <= 0 , b <= 0 } **12.2**

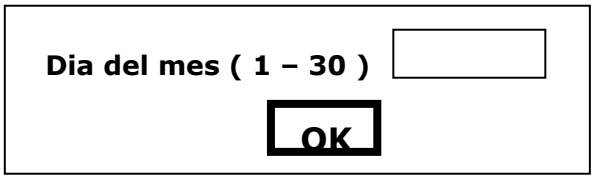
En la tabla se muestran en **negrilla** las expresiones que definen la partición derivada del tratamiento diferenciado en el módulo, y que por lo tanto se mantienen tanto en las clases "correctas" como en las "erróneas". La clase 9 se expresaría en realidad como { a > b; a > 0; b > 0 }, de ahí la conversión en dos clases erróneas (10.1 y 10.2), pero se simplifica a la forma en que aparece en la tabla ya que b > 0 y a > b implican el cumplimiento de a > 0 (me refiero a la clase 9). Lo mismo sucede con la clase 11. De la tabla de particiones obtenemos la siguiente batería de pruebas:

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	(3 , 3)	4	1 , 3 , 5, 7	(1 , 1)	1
	(27 , 12)	3	1 , 3 , 5, 9	(2 , 1)	1
	(4 , 6)	2	1 , 3 , 5, 11	(1 , 2)	1
Clases erróneas	( )	-1	2.1	(1)	-1
	(2 , 2 , 2 , 2)	-1	2.2	(2, 2, 2)	-1
	(4.71 , 2)	-1	4.1		
	(2 , 4.71)	-1	4.2		
	('a' , 2)	-1	4.3		
	(2 , 'a')	-1	4.4		
	(-4 , -4)	-1	6.1 , 6.2 , 8	(0 , 0)	-1
	(6 , -4)	-1	6.2 , 10.1	(1 , 0)	-1
	(-2 , -4)	-1	6.1 , 6.2 , 10.2	(0 , -1)	-1
	(-4 , 6)	-1	6.1 , 12.1	(0 , 1)	-1
	(-4 , -2)	-1	6.1 , 6.2 , 12.2	(-1 , 0)	-1

Los casos obtenidos en el análisis de valores límite se derivan de elegir valores adicionales lo más cercanos posibles a las condiciones de las clases. Por ejemplo, dada la condición a > b, se escoge un caso de prueba donde a = b+1 (suponiendo que tengamos ya un caso de prueba con a > b+1).

05. En un colegio reciben pagos de pensiones cada mes.
• Si la persona paga entre el día 1 y 10 no tiene ningún recargo.
• Si paga entre el 11 y 20 tiene un recargo del 2%.
• Si paga el día 21 o después el recargo es del 4%.

Se ha realizado un programa que sólo solicita el día del mes en que se paga (en este ejercicio el mes no importa, ya que todos los meses se consideran de 30 dias) y despliega el porcentaje de recargo a cobrar. Se anexa a continuación la pantalla:



Si no se introduce un valor acorde a lo descrito, el módulo devolverá el valor **"error"**. Diseñe los casos de prueba.

SOLUCION. Analizamos posibles entradas de datos:

Clases de datos (partición equivalente)

Válidos : Positivo → Un valor entre 1-10 - Un valor entre 11-20 - Un valor entre 21-30
No Válidos: Positivo fuera del rango especificado - Alfabéticos y Especiales - Nulo - Negativo - Cero - Flotante.

Valores límite a tener en cuenta: Rango → 0, 1, 10, 11, 20, 21, 30, 31.

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 1 } **1**	{ n < 1 (nulo) } **2**
	B	Tipo de los parámetros	{ dia ∈ N } **3** (entero)	{ dia ∉ N } **4.1** { dia ∈ Caracteres } **4.2** (alfabéticos y especiales)
A , B	C	Sin recargo (SR)	{ dia > 0 , dia <= 10 } **5**	{ dia < 0 } **6.1** (negativo) { dia = 0 } **6.2** (cero)
A , B	D	Recargo del 2% (R2)	{ dia >= 11 , dia <= 20 } **7**	
A , B	E	Recargo del 4% (R4)	{ dia >= 21 , dia <= 30 } **8**	{ dia > 30 } **9**

De la tabla de particiones obtenemos la siguiente batería de pruebas:

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	(4)	SR	1 , 3 , 5	(1) (10)	SR
	(18)	R2	1 , 3 , 7	(11) (20)	R2
	(26)	R4	1 , 3 , 8	(21) (30)	R4
Clases erróneas	( )	Error	2		
	(3.8)	Error	4.1		

	('ab')	Error	4.2		
	(-1)	Error	6.1		
	(0)	Error	6.2		
	(32)	Error	9	(31)	Error

06. Un subprograma tiene como entrada tres parámetros enteros (puede ser positivo, negativo o cero) x, y, z. Los valores de x e y debe representar un intervalo **[x , y]** donde:

"x" debe ser el extremo inferior **"y"** debe ser el extremo superior

Y el subprograma tiene como misión estudiar la pertenencia del valor z al intervalo. Las salidas posibles del programa son:
Extremo: Si z coincide con uno de los extremos del intervalo.
Medio: Si z es el punto medio del intervalo.
Interior: Si z pertenece al intervalo, pero no en las situaciones anteriores.
Exterior: Si z no pertenece al intervalo.
Error: Si la entrada es errónea (ya sea porque se introduce flotantes o caracteres).

SOLUCION. Hay una condición que no se mencionó pero que es muy importante y es el hecho que **x** debe ser menor que **y** puesto que de lo contrario, no se evalúa el intervalo. La ventaja de la situación planteada en este ejercicio es que cuando los valores son correctos, necesariamente deben pertenecer a uno de los cuatro casos (extremo, medio, interior o exterior).

Asume		**Condición**	**Clases correctas**	**Clases erróneas**
	A	N° de parámetros	{ n = 3 } **1**	{ n < 3 } **2.1** { n > 3 } **2.2**
	B	Tipo de los parámetros	{ x ∈ Z , y ∈ Z , z ∈ Z } **3**	{ x ∉ Z } **4.1** { y ∉ Z } **4.2** { z ∉ Z } **4.3**
	C	Orden intervalo	{ x < y } **5**	{ x > y } **6.1** { x = y } **6.2**
A , B , C	D	Extremo	{ z = x } **7.1** { z = y } **7.2**	
A , B , C	E	Medio	{ z = ((x + y) / 2) } **8**	
A , B , C	F	Interior	{ z > x , z < y , z <> ((x + y) / 2) } **9**	
A , B , C	G	Exterior	{ z < x } **10.1** { z > y } **10.2**	

	Entradas	**Salidas**	**Clases Cubiertas**	**Valores Límite**	**Salidas**
Clases correctas	(3 , 5 , 3)	Extremo	1 , 3 , 5 , 7.1		
	(3 , 5 , 5)	Extremo	1 , 3 , 5 , 7.2		
	(3 , 9 , 6)	Medio	1 , 3 , 5 , 8		
	(3 , 9 , 4)	Interior	1 , 3 , 5 , 9		
	(3 , 9 , 1)	Exterior	1 , 3 , 5 , 10.1		
	(3 , 9 , 14)	Exterior	1 , 3 , 5 , 10.2		
Clases erróneas	(8)	Error	2.1	(8 , 15)	Error
	(8 , 11 , 15 , 20 , 22)	Error	2.2	(2 , 7 , 10 , 14)	Error
	(3.1 , 9 , 14)	Error	4.1		
	('d' , 9 , 14)	Error	4.1		
	(3 , 9 .1 , 14)	Error	4.2		
	(3 , 'd' , 14)	Error	4.2		
	(3 , 9 , 14.1)	Error	4.3		
	(3 , 9 , 'd')	Error	4.3		
	(9 , 3 , 14)	Error	6.1		
	(9 , 9 , 14)	Error	6.2		

RECAPITULACION SOBRE PRUEBAS DE CAJA NEGRA

Hemos visto que hay 2 herramientas muy útiles para diseñar casos de prueba de caja negra, una de ellas es la partición equivalente y la otra es el análisis de valores límite.

A. Diseño de casos de prueba (para partición equivalente)

PASO 1 : IDENTIFICACIÓN DE LAS CLASES DE EQUIVALENCIA. Se identifican dividiendo en dos o más grupos (clases) cada condición de entrada, la cual es :

*Un valor numérico específico *Un rango de valores *Un conjunto de valores relacionados *Una condición booleana

Las clases de equivalencia se pueden definir de acuerdo a las siguientes directrices:

- Si una condición de entrada especifica un rango de valores (por ejemplo : valor entre 1 y 999), se definen una clase de equivalencia válida (1<=valor<=999) y dos inválidas (valor<1 y valor>999).

- Si una condición de entrada requiere un valor específico, (el tamaño de un código es de 5 dígitos), se definen una clase de equivalencia válida (tamaño del código 5) y dos inválidas (tamaño<5 y tamaño>5).

- Si una condición de entrada especifica un conjunto de valores de entrada (tipo de vehículo: AUTOBÚS, TAXI, TURISMO, MOTOCICLETA), se define una clase de equivalencia válida para cada valor y una clase de equivalencia inválida (por ejemplo PATINES).

- Si una condición de entrada es booleana (el primer carácter de un identificador debe ser una letra), se definen una clase válida (es letra) y una inválida (no es letra).

PASO 2 : IDENTIFICACIÓN DE LOS CASOS DE PRUEBA.

- Asignar un número único a cada clase de equivalencia.

- Escribir casos de prueba hasta que sean cubiertas todas las clases de equivalencia válidas, intentando cubrir en cada caso tantas clases de equivalencia válidas como sea posible.

- Escribir casos de prueba hasta que sean cubiertas todas las clases de equivalencia inválidas cubriendo en cada caso una y sólo una clase de equivalencia aún no cubierta.

B. Criterios para la generación de casos de prueba (para análisis de valores límite)

- Si una condición de entrada especifica un rango de valores limitado por a y b, se deben escribir casos de prueba para los límites del rango, es decir, a y b, y casos de prueba inválidos para situaciones fuera de los límites. Ej: si el rango válido de un valor de entrada es -1.0 a +1.0, escribir casos de prueba para las situaciones: -1.0, +1.0, -1.0001, +1.0001

- Si una condición de entrada especifica un número de valores, desarrollar casos de prueba para el número máximo y mínimo de valores y otros casos para los valores justo por encima y justo por debajo del máximo y del mínimo. Ej: si el número de alumnos para calcular su nota final es de 125, escribir casos de prueba para 0, 1, 125 y 126 alumnos.

- Aplicar el primer criterio para cada condición de salida. Por ejemplo, si un programa calcula la deducción fiscal mensual, siendo el rango permitido entre 0 y 1000 €. escribir casos de prueba que proporcionen el resultado de 0 € y 1000 €. Comprobar si es posible diseñar casos de prueba que puedan proporcionar una deducción negativa o mayor de 1000 €.

- Es importante examinar los límites del espacio de resultados pues no siempre ocurre que los límites de los dominios de entrada representan el mismo conjunto de circunstancias que los límites de los rangos de salida. (por ejemplo la función seno).

- Si las estructuras de datos internas tienen límites preestablecidos, hay que asegurarse de diseñar casos de prueba que ejerciten la estructura de datos en sus límites. Por ejemplo, un array que tenga un límite definido de 100 elementos, habrá que probar el hecho de que no existan elementos y el de que se procesen bien 100 elementos.

- Usar el ingenio para buscar otras condiciones límite (conjetura de errores).

FIN DEL DOCUMENTO