Ingeniería del Software

EJERCICIOS

Unidad 6: Pruebas

EJERCICIO 1

Dados los siguientes datos de entrada para un formulario de datos:

- DNI es un campo de números enteros positivos de 8 dígitos, mayor de 999999.
- Nombre es un campo alfanumérico de 25 caracteres exactamente.
- Día libre de la semana es un campo que puede tomar los valores (lunes, martes, miércoles, jueves, viernes, sábado, domingo).
- Antigüedad en la empresa en meses es un campo de dos dígitos que puede tomar el valor 0.

El programa asigna 4 ayudas diferentes en función de los días libres y de la antigüedad en la empresa de la siguiente forma:

- A1: Lunes y personas con antigüedad inferior a 1 año.
- A2: Martes o miércoles y personas con antigüedad inferior a 3 años.
- A3: Jueves o viernes y personas con antigüedad inferior a 8 años.
- A4: Sábado o domingo y personas con antigüedad superior o igual a 8 años.

a) Crea una tabla de clases de equivalencia en la que se indiquen:
   - Dato de entrada o atributo que se analiza
   - Clases válidas para ese dato de entrada
   - Clases no válidas para ese dato de entrada

	Clases de equivalencia	
Atributo	Válida	No válida
DNI	1. [01000000 .. 99999999]	2. < 1000000 3. 99999999 4. No es número
NOMBRE	5. Cualquier composición alfanumérica de 25 caracteres	6. Cadena < 25 caracteres 7. Cadena > 25 caracteres 8. Carácter distinto de alfanumérico
DIA LIBRE	9. {Lunes, Martes, Miercoles, Jueves, Viernes, Sábado, Domingo}	10. Cualquier otra cadena
ANTIGÜEDAD	11. [0..99]	12. > 99 13. No es número

b) Genera los casos de prueba usando la técnica de particiones de equivalencia.

CASOS VÁLIDOS

Salida : Ayuda A1 Datos de Entrada: DNI: 8.978.765 – NOMBRE: "Antonio Rodríguez........" – Día Libre: Lunes – Antigüedad: 5 Clases válidas: 1, 5, 8, 10
Salida: Ayuda A2 Datos de Entrada: DNI: 2.728.765 – NOMBRE: "Fernando Rodríguez......." – Día Libre: Martes – Antigüedad: 13 Clases Válidas: 1, 5, 8, 10
Salida: Ayuda A3 Datos de Entrada: DNI: 5.728.765 – NOMBRE: "Martina Rodríguez........" – Día Libre: Viernes – Antigüedad: 40 Clases Válidas: 1, 5, 8, 10
Salida: Ayuda A4 Datos de Entrada: DNI: 9.728.765 – NOMBRE: "Enrique Rodríguez........" – Día Libre: Domingo – Antigüedad: 99 Clases Válidas: 1, 5, 8, 10
CASOS NO VÁLIDOS
Datos de Entrada: DNI: 0 – NOMBRE: "Pepe" – Día Libre: X – Antigüedad: 999 Clases Válidas: 2, 6, 9, 11
Datos de Entrada: DNI: 9999999999 – NOMBRE: "Pepe Pepe Pepe Pepe Pepe Pepe Pepe" – Día Libre: X – Antigüedad: AAA Clases Válidas: 3, 7, 9, 12
Datos de Entrada: DNI: BCDDDD – NOMBRE: "Pepe Pepe Pepe Pepe Pepe Pepe Pepe" – Día Libre: X – Antigüedad: AAA Clases Válidas: 4, 7, 9, 12

Etc.

EJERCICIO 2

Dada la siguiente descripción de un caso de uso, definir los casos de prueba utilizando el método de la partición equivalente.

Conteste en el espacio reservado para ello, dentro de este mismo enunciado.

Caso de uso: Sacar dinero
Actores: Cliente
Objetivo: Sacar dinero del cajero
Descripción: Un Cliente solicita realizar una operación de reintegro por una cantidad específica. El cajero le da el dinero solicitado tras comprobar que la operación puede realizarse. El Cliente coge la tarjeta, el recibo, el dinero y se va.
Referencias cruzadas: R1.3, R1.7
Flujo de eventos:

ACCIÓN DEL ACTOR	RESPUESTA DEL SISTEMA
1. Selecciona la operación de Reintegro	2. Pide la cantidad a retirar.
3. Introduce la cantidad requerida	4. Procesa la petición y da el dinero solicitado. Devuelve la tarjeta y genera un recibo
5. Recoge la tarjeta	

6. Recoge el recibo	
7. Recoge el dinero y termina el caso de uso	

Caminos alternativos:
- Evento 4: La cantidad solicitada supera el saldo. Se indica el error y se cancela la operación.
- Evento 4: La cantidad solicitada supera el límite diario. Se indica el error y se cancela la operación.

a) Crear una tabla de clases de equivalencia en la que se indiquen:
- Dato de entrada o atributo que se analiza
- Clases válidas para ese dato de entrada
- Clases no válidas para ese dato de entrada

	Clases de equivalencia	
Atributo	**Válida**	**No válida**
Saldo	1. >0	2. ≤0
Cantidad pedida	3. ≤ saldo & ≤ límite diario	4. >saldo 5. >límite diario
Dinero devuelto	6. Cantidad pedida	7. 0

b) Definir las combinaciones de las clases anteriores necesarias para cubrir todas las clases de equivalencia. Puede añadir las filas que necesite.

Escenario	**Saldo**	**Cantidad pedida**	**Cantidad pedida**
Camino básico	Positivo	cantidad pedida ≤ saldo	cantidad pedida ≤ límite diario
No hay saldo	Positivo	cantidad pedida > saldo	cantidad pedida ≤ límite diario
Se excede el límite diario	Positivo	cantidad pedida ≤ saldo	cantidad pedida > límite diario

Si el saldo es negativo la cantidad pedida siempre será superior al saldo por lo que se trata del caso "No hay saldo"

c) Definir los casos de prueba que cubren las particiones anteriores. Puede añadir las filas que necesite.

Escenario	**Saldo**	**Límite diario**	**Cantidad pedida**	**Dinero entregado**
Camino básico	3100,25	450	300	300
	-12,47	450	1	0
	0	450	1	0

No hay saldo	50,32	450	51	0
Se excede el límite diario	685,96	450	451	0

EJERCICIO 3

Dado el siguiente fragmento de código:

```
If (h>=0) and (h<=23) then
  if (m>=0) and (m<=59) then
    if (s>=0) and (s<=59) then
        Then write ('La hora es correcta')
        Else write ('La hora no es correcta')
```

SE PIDE:

Genera los casos de prueba necesarios para obtener una cobertura completa de decisión/condición.

En el código hay tres decisiones:
D1 -> (h>=0) and (h<=23)
C1.1 -> h>=0
C1.2 -> h<=23
D2 -> (m>=0) and (m<=59)
C2.1 -> m>=0
C2.2 -> m<=59

D3 -> (s>=0) and (s<=59)
C3.1 -> s>=0
C3.2 -> s<=59

Hay que garantizar que cada condición tome al menos una vez el valor verdadero y otra el valor falso, garantizando además que se cumpla la cobertura de decisión. Los datos concretos para los casos de prueba podrían ser los siguientes:

	Valor Verdadero	Valor Falso
C1.1	h=12	h=-1
C1.2	h=12	h=24
C2.1	m=45	m=-1
C2.2	m=45	m=60
C3.1	s=33	s=-1
C3.2	s=33	s=60

Si tomamos los datos de C11 y C12 que hacen que tomen los valores VERDADERO simultáneamente, la decisión D1 tomará también el valor VERDADERO. Para que tome el valor FALSO, no podemos hacer que C11 y C12 tomen los valores FALSO simultáneamente, y habrá que considerar dos casos: C11= V, C12 = F y C11=F, C12=V para que la decisión D1 tome también el valor FALSO cubriendo todas las condiciones.

```
Lo mismo ocurre con C2.1 y C2.2 y con C3.1 y C3.2.

Caso de prueba 1: C1.1= Verdadero, C1.2= Verdadero, C2.1= Verdadero, C2.2=
                  Verdadero, C3.1= Verdadero, C3.2= Verdadero
      h=12; m=45; s=33

Caso de prueba 2: C1.1= Verdadero, C1.2= Verdadero, C2.1= Verdadero, C2.2=
                  Verdadero, C3.1= Verdadero, C3.2= Falso
      h=12; m=24; s=60

Caso de prueba 3: C1.1= Verdadero, C1.2= Verdadero, C2.1= Verdadero, C2.2=
                  Verdadero, C3.1= Falso, C3.2= Verdadero
      h=12; m=24; s=-1

Caso de prueba 4: C1.1= Verdadero, C1.2= Verdadero, C2.1= Verdadero, C2.2= Falso
      h=12; m=60

Caso de prueba 5: C1.1= Verdadero, C1.2= Verdadero, C2.1= Falso, C2.2= Verdadero
      h=12; m=-1

Caso de prueba 6: C1.1= Verdadero, C1.2= Falso
      h=24

Caso de prueba 7: C1.1= Falso, C1.2= Verdadero
      h=-1
```

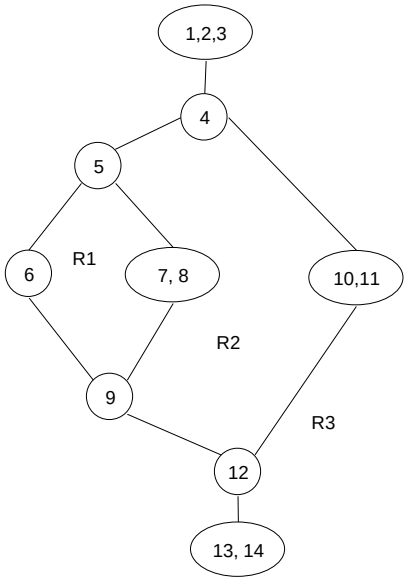
EJERCICIO 4

Dado el siguiente fragmento de código de impresión de tarjetas de personal:

```
1   Begin
2       Lee DatosPersona
3       CodigoBarras = CodigoBarras + 1
4       if Edad >= 30 and Persona = "VIP" then
5               if Cargo = "Director" then
6                       ColorTarjeta = “Dorado”
7               else
8                       ColorTarjeta = “Rojo”
9               endif
10      else
11              ColorTarjeta = “Blanco”
12      endif
13      PrintTarjeta CodigoBarras, Nombre, FechaNacimiento, ColorTarjeta
14  End
```

a) Determinar por el método de McCabe el número de caminos básicos encontrados para el código anterior, de dos formas diferentes.

```
Se numeran las sentencias del código del enunciado y se realiza el grafo
siguiente:
```



NÚMERO DE CAMINOS BÁSICOS:
A – N + 2 = 10 – 9 + 2 = 3
n° de regiones cerradas + 1 = 2 (R1, R2) + 1 = 3

b) Genera los casos de prueba necesarios para obtener una cobertura completa de decisión/condición.

En el código del enunciado hay dos decisiones:
D1 -> "Edad >= 30 and Persona = 'VIP'"que está formada por dos condiciones:
C11 -> Edad >= 30
C12 -> Persona = 'VIP'
D2 -> "Cargo = 'Director'"

Hay que garantizar que cada condición tome al menos una vez el valor verdadero y otra el valor falso, garantizando además que se cumpla la cobertura de decisión. Los datos concretos para los casos de prueba podrían ser los siguientes:

		Valor Verdadero	Valor Falso
D1	**C11**	Edad = 32	Edad = 25
	C12	Persona = 'VIP'	Persona = 'noVIP'

Si tomamos los datos de C11 y C12 que hacen que tomen los valores VERDADERO simultáneamente, la decisión D1 tomará también el valor VERDADERO. Si tomamos los datos de C11 y C12 que hacen que tomen los valores FALSO simultáneamente, la decisión D1 tomará también el valor FALSO. De esta forma se garantiza la cobertura de decisión/condición con el mínimo número de casos de prueba posibles (2 casos de prueba).
Si además consideramos el caso de la D2 del apartado c), tenemos los siguientes 3 casos de prueba:

Caso de prueba 1: D1= Verdadero (C11 = Verdadero = C12); D2 = Verdadero
Edad = 32; Persona = 'VIP'
Cargo = 'Director'
Caso de prueba 2: D1 = Falso (C11= Falso = C12); D2= Indiferente

```
  Edad = 25; Persona = 'noVIP'
  Cargo = valor INDIFERENTE
Caso de prueba 3: D1= Verdadero (C11 = Verdadero = C12); D2= Falso
  Edad = 32; Persona = 'VIP'
  Cargo = 'Conserje'
```

EJERCICIO 5

Dada la siguiente función "C":

```
#include <stdio.h>
#include <math.h>

#define PRECISION 1e-6

double calculaRaizCuadrada ( double num )
/* calcula la raiz cuadrada de un número positivo iterativamente */
{
   double sup, inf, raiz, division;
```

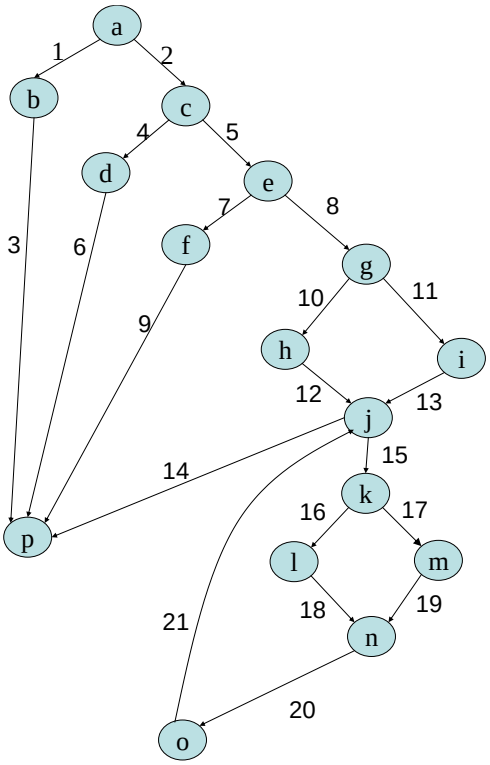
a	`if (num<0)`		
b	`raiz = -1;`		
c	`else if (num==0)`		
d	`raiz = 0;`		
e	`else if (num==1)`		
f	`raiz = 1;`		
	`else`		
	`{`		
g	`if ( num > 1 )`		
	`{`		
h	`sup  = num/2.0;` `raiz = num/3.0;`		
	`}`		
	`else`		
	`{`		
i	`sup = 1.0;` `raiz = 0.5;`		
	`}`		

j	`inf  = 0.0;`
	`division = num/raiz;`
	
	`while (fabs(raiz-division)>PRECISION)`		

	`{`		
k	`if (raiz*raiz > num)`	**l**	`sup = raiz;`
m	`else inf = raiz;`		
n	`raiz = (sup+inf)/2.0;` `division = num/raiz;`		
o	`}`		
	`}`		
p	`return raiz;`		
	`}`		

a) Dibuja el grafo de flujo ¿Cuál es la complejidad ciclomática?

```
Aristas - Nodos + 2 = 21 - 16 + 2 = 7
```



b) ¿Cuáles son los caminos básicos?

```
Caso 1: 1-3
Caso 2: 1-2-4-6
Caso 3: 1-2-5-7-9
Caso 4: 1-2-5-8-10-12-14
Caso 5: 1-2-5-8-11-13-14
Caso 6: 1-2-5-8-10-12-15-16-18-20-21-14
Caso 7: 1-2-5-8-11-13-15-17-19-20-21-14
```

c) Prepara un conjunto de casos de pruebas que ejerciten todos los caminos básicos.

```
Caso 1: num= -4, raiz=-1
Caso 2: num= 0, raiz=0
Caso 3: num= 1, raiz=1
Caso 4: num= 9, raiz=3          (sup=9/2, raiz=9/3, division=3)
Caso 5: num= 0.25, raiz=0.5           (num/0,5=0,5 => num=0,25)
Caso 6: num= 36, raiz=6         (sup=n/2,raiz=n/3, div=n/3,
                                 bucle::raiz=(n/3)/2=n/6,
                                Div=n/(n/6)=6,  n/6-6=0 => n=36)

Caso 7: num=0.5625, raiz=0.75
       (Sup=1.0, raiz=0,5, inf=0, div=num/0.5    condición:: 0.5*0.5 <= num
       Bucle:: inf=0.5, raiz=(1+0.5)/2=0.75      , div=num/0.75 =>
       0.75-num/0.75=0   => num=0.75²)
```

d) ¿Qué otras pruebas de caja blanca añadirías? ¿Por qué? Pon un ejemplo enumerando algunos de estos casos.

Harían falta pruebas de bucles. Con las pruebas del camino básico hemos hecho 0 y 1 pasadas. Habría que seleccionar casos de prueba que hicieran n pasadas (un valor medio), y un número muy alto de pasadas. También habría que probar con números menores que uno. Además se pueden probar con números muy pequeños (menores que PRECISION), o números muy grandes.