Escuela Politécnica Superior

UAM Universidad Autónoma de Madrid

Ingeniería del Software

EJERCICIOS

Unidad 5: 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

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

EJERCICIO 2

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

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

b) Definir las combinaciones de las clases anteriores necesarias para cubrir todas las clases de equivalencia.

c) Definir los casos de prueba que cubren las particiones anteriores.

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:

a. Generar los casos de prueba necesarios para obtener una cobertura completa de decisiones.

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

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.

b) Genera los casos de prueba necesarios para obtener una cobertura completa de decisiones.

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

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?

b) ¿Cuáles son los caminos básicos?

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

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