Testing funcional

Ejercicio 1:

Dada la siguiente especificación:

Un programa toma como entrada un fichero cuyo formato de registro es el siguiente:

Numero-empleado	Nombre-empleado	Meses-Trabajo	Directivo

-Numero-empleado es un campo de números enteros positivos de 3 dígitos (excluido el 000).
-Nombre-empleado es un campo alfanumérico de 10 caracteres.
-Meses-Trabajo es un campo que indica el número de meses que lleva trabajando el empleado; es un entero positivo (incluye el 000) de 3 dígitos.
-Directivo es un campo de un solo carácter que puede ser «+» para indicar que el empleado es un directivo y «-» para indicar que no lo es.

El programa asigna un premio (que se imprime en un listado) a cada empleado según las normas siguientes:
o P1 a los directivos con, al menos, 12 meses de antigüedad
o P2 a los no directivos con, al menos, 12 meses de antigüedad
o P3 a los directivos sin un mínimo de 12 meses de antigüedad
o P4 a los no directivos sin un mínimo de 12 meses de antigüedad

a) Desarrollar la estrategia de **Clases de Equivalencia**:
(i) Crear una Tabla de Clases de Equivalencia (las clases deberán ser numeradas) en la que se indiquen las siguientes columnas en cada fila:
- Condición de entrada que se analiza
- Clases válidas y
- Clases no válidas que se generan para la condición
(ii) Generar los casos de prueba (especificando la entrada en todos los casos y la salida esperada sólo en los casos válidos) para las clases creadas, indicando en cada caso las clases que cubre.

Ejercicio 2:
Dada una pantalla de datos con los siguientes campos:

- Nombre: Texto, hasta 50 caracteres. No puede ser vacío.
- Millas acumuladas: Un float mayor o igual a cero (el control en la UI no permite el ingreso de valores no numéricos)
- Tipo de pasajero: {Viajero frecuente, Corporativo, Turista, VIP} El pasajero VIP recibe un tratamiento especial. (El control es un ListBox, una lista.
- Edad: Un valor entero entre 0 (inclusive) y 100 (inclusive)

Diseñar casos de test para la aplicación de ejemplo con el método de Equivalence Classes.
Aumentar lo diseñado en el punto 1 con la técnica de Boundary Values.

Ejercicio 3:
Diseñar casos de test utilizando la técnica de tablas de decisión. Para otorgar descuentos en una cadena de cines, existen las siguientes bases: los menores de edad y jubilados, cuentan con descuento de 50% todos los días de la semana, en todas las funciones. Para el resto de los espectadores, el descuento es de 50% los días miércoles. Adicionalmente, los menores en vacaciones de invierno, cuentan con un descuento adicional del 20% en las funciones hasta las 20hs.

Ejercicio 4:
Para los cursos particulares que realice el colaborador, técnicos y que involucran una certificación, la empresa X contribuye con un 40% del valor. Si no involucran una certificación, X contribuye en este tipo

de cursos con 30% de su valor. Para los no técnicos, X contribuye con 30% para los que involucran certificaciones y 20% para los que no involucran certificaciones.
En cuanto a los cursos corporativos, si involucran certificaciones, el colaborador se hace cargo siempre de 20% del costo. En Argentina el estado permite deducir de impuestos el 10% del costo de los cursos corporativos técnicos.
La aplicación refleja esta información en el campo CostoFinal, que refleja el costo que tiene el curso para la organización.
Diseñar casos de test para este escenario con el método de Decision Tables.

Ejercicio 5:
Diseñar un conjunto de casos de test **minimal (no puedo sacar ningún caso de test, ni ninguna parte de un caso de test y obtener el mismo resultado)** con el criterio de 'todas las transiciones', para el siguiente diagrama de estados:

Ejercicio V

Supongamos que la aplicación debe ser testeada utilizando

- Internet Explorer 5.5, 6, 7 y 8, Firefox 2 y 3
- Flash Player 7, 8, 9 y 10
- WinXP, Windows Vista, Windows 2000
- Base de datos Oracle, SQLServer, DB2, PostgreSQL, mySQL, Apache Derby
- Servidor de aplicaciones WebSphere, JBoss, Glassfish, Tomcat

No tenemos tiempo para probar todas estas combinaciones. Elegir qué combinaciones vamos a testear utilizando el método de Pairwise Testing.