LETRA A

1. De los principios de diseño vistos en el curso
   a) el de modularización plantea descomponer el software en varias piezas para manejar la complejidad del mismo y localizar el impacto de los cambios
   b) a) y para obtener una buena modularización se aplican también los principios de alta cohesión y bajo acoplamiento que indican que cada módulo debe realizar tareas relacionadas, minimizando la interacción con el resto
   c) b) y el de separación de intereses indica como separar funcionalidades que deban ser provistas por un mismo módulo, de forma de no contravenir los principios de alta cohesión y bajo acoplamiento
   d) Ninguna de las anteriores

2. Un patrón de diseño provee una solución general a un problema en un contexto
   a) que se compone de un nombre, la descripción del problema que resuelve, la solución provista y las consecuencias de su aplicación, en cuanto a la tecnología utilizada
   b) a) y se provee también un diagrama de clases que indica las clases, atributos y métodos de cada una, y la interacción entre éstas
   c) permitiendo entre otras cosas la reutilización de diseños, de la misma forma que al utilizar un framework se reutilizan tanto el diseño como el código
   d) c) y hay frameworks que implementan patrones de diseño, por ejemplo el framework Struts con el patrón MVC (Model View Controller)

3. En cuanto al tratamiento de anomalías se tienen distintas opciones, donde
   a) la opción de reintentar plantea restaurar el sistema e intentar nuevamente con la misma estrategia
   b) la opción de corregir plantea restaurar el sistema, corregir algo e intentar nuevamente con otra estrategia
   c) la opción de informar plantea avisar a alguien de la imposibilidad e intentar nuevamente con alguna de las opciones anteriores
   d) ninguna de las anteriores

4. Entre las técnicas vistas para evitar fallas se encuentran
   a) revisar periódicamente síntomas de fallas mediante la ejecución en paralelo de otros procesos que controlen diversos aspectos de la ejecución del sistema
   b) sospecha mutua donde todos los componentes interrogan al resto de los componentes con los que interactúan para controlar que la ejecución de cada uno sea correcta
   c) procesos independientes sincronizados donde se plantea ejecutar varios procesos que comparan resultados continuamente para asegurar la correcta ejecución de los componentes
   d) ejecutar n versiones iguales de los componentes que realizan las mismas tareas y eligen la acción siguiente en la ejecución mediante la implementación de un sistema de votación

5. En las pruebas de caja blanca
   a) se crean los casos de prueba antes de la implementación del módulo a probar.
   b) se obtienen los datos de prueba únicamente a partir del código fuente y usando la especificación se obtiene el resultado esperado; con lo cual se tienen los casos de prueba.
   c) se obtienen los casos de prueba únicamente a partir del código fuente.
   d) (c) y son apropiadas para definir los casos de prueba a usar en una prueba de desempeño del sistema

6. El objetivo principal de la verificación es:
   a) Descubrir defectos
   b) Corregir defectos
   c) Estipular junto con el cliente los criterios de aceptación de un producto
   d) Ninguna de las anteriores

7. Según el proceso de verificación dado en el curso, ¿cuál de estas pruebas **no** es de desempeño?:
   a) Pruebas de volumen
   b) Pruebas de regresión
   c) Pruebas de seguridad
   d) Todas las pruebas mencionadas son de desempeño

8. Respecto a cuándo es aconsejable detener las pruebas
   a) Si se dejan de provocar fallas durante el testing se deben detener las pruebas
   b) Se deben detener las pruebas si el tiempo planificado para las mismas ha expirado
   c) Las pruebas de sistema se detienen cuando todos los casos de prueba que fueron creados durante el análisis de requerimientos han sido ejecutados
   d) Ninguna de las anteriores

9. Dado un programa con la siguiente decisión:
   if (a < b) then SentenciaConDefecto
   ¿Cuál de estos criterios de cubrimiento asegura que siempre se va a provocar una falla con un conjunto de casos de prueba que satisface el criterio?
   a) Criterio de cubrimiento de sentencias
   b) Criterio de cubrimiento de decisión/condición
   c) Criterio cubrimiento de todas las definiciones
   d) Ninguna de las anteriores

10. Dado el siguiente fragmento de programa
```
if a < b and b > 0
    if  (a <> 0)
        return a div b;
    else     return b div a;
else     return 1;
```
   en un lenguaje en el cual si se ejecuta una división por cero se aborta la ejecución se pregunta:
   ¿Cuál de estos criterios de cubrimiento es el menos fino que asegura que siempre se va a provocar una falla al ejecutar una división entre cero?
   a) Criterio de cubrimiento de sentencias
   b) Criterio de cubrimiento de decisión/condición
   c) Criterio de cubrimiento de condición múltiple
   d) Ninguna de las anteriores

11. ¿Cuál de las siguientes técnicas no es una técnica de testing de caja negra?
   a) Partición en clases de equivalencia
   b) Mutation testing
   c) Análisis de valores límite
   d) Grafos causa-efecto

12. No es una tarea de la liberación:
   a) Capacitación a los usuarios
   b) Conversión (migración) de datos desde otro sistema
   c) Soporte a usuarios y solución de problemas
   d) Prueba de integración y del sistema

13. El momento más adecuado para comenzar a planificar las tareas de la liberación de un sistema es:
   a) Al inicio del proyecto, durante la definición del concepto de operación
   b) Al contar con requerimientos estables y aprobados por usuarios y cliente
   c) Al contar con una arquitectura estable
   d) Al contar con un producto estable

14. El mantenimiento de software:
   a) Es una tarea que requiere poco esfuerzo y que prácticamente no tiene importancia económica para las organizaciones de desarrollo de software
   b) Correctivo es aquél destinado a prevenir defectos en el software, esto es evitar que estos se manifiesten
   c) Perfectivo tiene por objetivo generar productos sin defectos (cero defecto)
   d) Es más o menos difícil de llevar a cabo, dependiendo de diversos atributos de calidad del software, tales como la moularidad y verificabilidad

15. Respecto al mantenimiento
   a) Una caída en el esfuerzo destinado al mantenimiento correctivo puede estar indicando problemas en la calidad del desarrollo o del mantenimiento
   b) Conviene establecer una política de liberación de versiones de un producto, para que resulte factible llevar a cabo un adecuado control de la calidad de cada liberación
   c) (b), en particular, para poder llevar a cabo una adecuada prueba de regresión
   d) (c), esa política debiera estar complementada por procedimientos de emergencia que permitan atender problemas que no pueden esperar hasta la próxima liberación planificada

16. Existen varios enfoques para evaluar productos de software que se enumeran a continuación:
   a) Estudios retrospectivos y análisis de características
   b) Análisis de riesgos y de tiempo medio entre fallas
   c) a) y estudios de casos y experimentos formales
   d) b) y encuestas de satisfacción

17. La calidad en el uso según la norma ISO 9126 se define como:
    a) Sea amigable y permita trabajar en forma efectiva y eficiente
    b) La capacidad de un producto de software de permitir que usuarios específicos cumplan objetivos de forma efectiva, productiva y segura
    c) a) y maneje correctamente las excepciones
    d) b) y satisfactoria en contextos de uso especificados

18. La capacidad de un proceso se define como:
    a) El rango de resultados esperado cuando se ejecuta el proceso
    b) a) y permite predecir el desempeño de futuras ejecuciones
    c) La medida de los resultados reales obtenidos mediante su ejecución
    d) a) y en un entorno de alta productividad

19. El mejor momento para realizar un análisis post-mortem de un proceso es:
    a) Enseguida de terminado porque en general puede participar todo el equipo
    b) Antes de culminar porque la experiencia está aún fresca en todos los participantes
    c) Un año después porque hay suficiente perspectiva e información recolectada
    d) No existe un mejor momento aplicable para todos los proyectos

20. Las categorías de áreas de proceso del modelo CMMI son:
    a) Ingeniería, control de la calidad, administración de los proyectos y administración de subcontratos
    b) Administración de los procesos, administración de los proyectos e ingeniería
    c) b) y soporte
    d) ninguna de las mencionadas

21. Ejercicio de Verificación (12 puntos)

Parte 1. (9 puntos)

El método merge de la clase ExtrangeMergeVector recibe tres vectores vecOrd1, vecOrd2 y vecOrdCual, y devuelve un vector resultado que es el merge de vecOrd1 y vecOrd2 con el siguiente criterio:

vecResultado(pos_i)	=	vecOrd1(pos_i)	**SI** vecOrdCual(pos_i) = 1 y predicado(pos_i)
		vecOrd2(pos_i)	**SI** vecOrdCual(pos_i) = 2 y predicado(pos_i)
		vecOrd1(pos_i)	**SI** pos_i < vecOrd1.size() y pos_i >= vecOrd2.size()
		vecOrd2(pos_i)	SI pos_i < vecOrd2.size() y pos_i >= vecOrd1.size()

predicado(i) = i < vecOrd1.size() y i < vecOrd2.size()

vecCual es un vector que contiene elementos de los cuales se puede obtener el número 1 o el número 2, a eso se refiere vecOrdCual(pos_i) = 1 o igual a 2. Asumir que este vector SOLO contiene **estos valores**. **Recordar que las posiciones en los vectores de Java comienzan en cero.**

Se sabe que vecResultado.size() = size mas grande entre vecOrd1 y vecOrd2

Se pide: dar casos de prueba para el método merge intentando cubrir la mayor cantidad posible de defectos que podría contener este método. Usar partición en clases de equivalencia y valores límite de esas clases. En caso que se desconozca el resultado esperado poner un signo de interrogación (?)

Parte 2. (3 puntos)

Un método generator de una clase RandomGenerator recibe dos parámetros: int min, int max y devuelve un número int **TOTALMENTE** al azar entre min y max (pudiendo ser este número tanto min como max).
public int generator(int min, int max)

Ahora, el método merge de la clase ExtrangeMergeVector cambia y su especificación es la siguiente:
public Vector merge(Vector vecOrd1, Vector vecOrd2)

El método usa la función generator con min = 1 y max = 2. Entonces, para cada posición del vector resultado que tenga que decidir entre un elemento de vecOrd1 o vecOrd2 llama al método generator y si lo que el método devuelve es 1 entonces elige de vecOrd1 y si es 2 elige de vecOrd2.

Se pide: a) Dar una estrategia de integración para integrar estas dos clases. Justificar.
b) Indicar cómo se realizaría la verificación en cada paso de la estrategia elegida. Justificar.

22. Ejercicio de Evaluación de productos, procesos y recursos (8 puntos)

Se desea evaluar un herramienta para reportar incidentes.

1. ¿Qué método de evaluación de productos utilizaría?
2. Describa las características deseables que debería tener la herramienta
3. Describa ventajas y desventajas del método de evaluación aplicado