- Cada pregunta de la **parte múltiple opción** contestada correctamente tiene un valor de 1,5 puntos. Cada pregunta incorrecta de la múltiple opción resta 0,5 puntos. Esta parte consta de 25 preguntas por lo que vale en total 37,5 puntos.
- El **ejercicio 1** vale 10,5 puntos. **Este ejercicio se debe contestar en la hoja MO usando las preguntas 26 a 28.**
- El **ejercicio 2** vale 12 puntos. **Este ejercicio se contesta en hoja especial para el mismo.**
- El puntaje total del parcial es 60 puntos.

Parte 1 Múltiple Opción

1) Analice el siguiente código que aparece a continuación.

```
1 boolean mayorEdad(int edad){
2     if (edad > 18){
3           return true;
4     }
5     return false;
6 }
```

¿Con qué criterio de cubrimiento se detectaría la falta que contiene? Seleccione el criterio menos fino (menos exigente).

a) Cubrimiento de sentencias.
b) Cubrimiento de condición.
c) Cubrimiento de decisión.
d) Ninguno de los anteriores

---

2) En lo que respecta al Diseño

a. Es una actividad básicamente creativa.
b. a) que no puede reducirse a una serie de pasos a seguir.
c. b) y, si se logra manejar la complejidad, se reducen los costos del diseño.
d. b) y si se logra manejar la complejidad no se reduce la posibilidad de introducir defectos durante el diseño.

---

3) La planificación de la liberación (puesta en explotación) de un software:

a. Debe comenzar al inicio del proyecto para manejar de forma adecuada el alcance del producto.
b. Conviene que comience al inicio del proyecto para gestionar de forma adecuada el alcance del proyecto.
c. Debe comenzar una vez que el producto está suficientemente estable (se dispone de una versión beta en condiciones de ser evaluada por usuarios y cliente) para que el esfuerzo de planificación se pueda desarrollar sobre bases ciertas.
d. Conviene que se lleve a cabo una vez que la arquitectura está estable, para que los requerimientos no funcionales estén claros.

---

4) Con respecto a la verificación:

a) Las pruebas unitarias solamente las puede realizar quien desarrolló la unidad que está bajo prueba.
b) En las pruebas de integración es de suma importancia conocer las interfaces del sistema.
c) Para las pruebas que no son unitarias y de integración, es conveniente que las realice un equipo especializado interno, no externo. Debido a que es necesario que el equipo de pruebas conozca los requerimientos y tenga una visión global.
d) b) y c) son correctas.

---

5) Con respecto a los Frameworks:

a. Es recomendable su uso ya que no requiere un gran esfuerzo de aprendizaje por tener un alto grado de aceptación.
b. a), y se puede definir como un subsistema reusable y semicompleto.
c. Presentan un nivel de abstracción de mayor granularidad que los objetos.
d. c), pudiendo ser extendido, por ejemplo mediante el desarrollo de clases concretas a partir de clases abstractas provistas en el Framework.

---

6) En auditoria de la configuración

a. Se busca verificar que en un momento dado, el sistema en desarrollo es una colección de productos consistente y bien definida.
b. Se previenen problemas.
c. La generación de informes es útil para brindar información referente a incidentes ocurridos.

d. Todas las anteriores.

---

7) Acerca de las estrategias de integración:
   a) Big-Bang es considerada una estrategia no incremental, mientras que Bottom-Up y Top-Down son consideradas estrategias incrementales.
   b) a) y en Big-Bang se prueba cada módulo de forma aislada y luego se prueba la combinación de todos los módulos a la vez.
   c) b) y en Bottom-Up se comienza por los módulos que requieren de otro(s) para ejecutar y se sigue hacia arriba según la jerarquía "usa".
   d) d) y en Top-Down se comienza por el módulo inferior de la jerarquía "usa" y se sigue hacia abajo según dicha jerarquía.

---

8) La conversión (sustitución) de un sistema ya existente por uno nuevo:
   a. A menudo plantea la necesidad de migrar datos del sistema anterior al nuevo. Esta migración en algunos casos resulta tan compleja que puede llegar a convenir tratarlo como un proyecto independiente paralelo (aunque relacionado) al desarrollo del sistema nuevo.
   b. Conviene, cuando sea posible, llevarla a cabo realizando un proceso en paralelo del sistema anterior y del nuevo, ya que es un excelente método de detección de defectos que hayan podido pasar inadvertidos por la prueba del sistema.
   c. Una de las mayores ventajas del procesamiento en paralelo es que dado que ambos sistemas (nuevo y viejo) se encuentran disponibles, su ejecución no requiere prácticamente de esfuerzo adicional.
   d. La sustitución de uno por otro en un instante en el tiempo (big-bang) generalmente es la estrategia que permite una mejora gestión de los riesgos inherentes a dicha sustitución.

---

9) Los diagramas de despliegue
   a. Identifican qué hardware usan los distintos componentes de software.
   b. a) e indicando qué clases son instaladas en cada nodo.
   c. a) y cada nodo es identificado de forma individual.
   d. c), pudiéndose tratar de dispositivos o ambientes de ejecución.

---

10) Respecto a las clasificaciones de faltas
   a) Las clasificaciones de faltas se basan en el tipo de falla que provoca una falta.
   b) a) y en los errores humanos que generan faltas.
   c) b) y clasifican faltas que se introducen únicamente en la fase de implementación.
   d) Sirven para evaluar a los implementadores de acuerdo a los tipos de faltas en las cuales incurren.

---

11) El mantenimiento de software se puede clasificar en "correctivo", "preventivo", "perfectivo" y "adaptativo". El esfuerzo que dedica una organización a cada uno de estos distintos tipos de mantenimiento puede considerarse un indicador de la calidad del software bajo mantenimiento y también del propio proceso de mantenimiento.
   a. En general es deseable que una organización de mantenimiento dedique la mayor parte de su esfuerzo al mantenimiento de tipo "correctivo".
   b. Un caso de mantenimiento "preventivo" se da cuando en una organización se decide reescribir un componente que presenta una historia de múltiples defectos reportados y ya corregidos a lo largo de un período.
   c. Un elevado esfuerzo promedio por solicitud de mantenimiento de tipo "adaptativo" puede estar indicando que el software que está siendo mantenido resulta fácil de adaptar a cambios en el ambiente o a nuevas necesidades.
   d. Un caso de mantenimiento "perfectivo" se da cuando en una organización se decide reescribir un componente que presenta una historia de múltiples defectos reportados y ya corregidos a lo largo de un período.

---

12) El principio abierto-cerrado
   a. Tiene como objetivo la construcción de sistemas con facilidad para su modificación.
   b. Es el principio de diseño más importante, ya que afecta fuertemente la mantenibilidad del sistema construido.
   c. a) y se entiende por "Abierto" que el comportamiento del sistema puede ser extendido.
   d. c) y se entiende por "Cerrado" que debe contarse con el código fuente para la realización de cambios.

---

13) Dentro de los objetivos de la verificación se tiene:
   a) Provocar fallas y revisar los productos para detectar defectos.
   b) a) y corregir estos defectos.

c) Evaluar y mejorar la calidad de los productos.
d) Todas las anteriores.

---

14) Los modelos de calidad del software son útiles por ejemplo para:
   a. Describir funcionalidades y atributos del software en los manuales de usuario.
   b. Definir los requerimientos de calidad de un producto de software.
   c. Evaluar un producto de software antes de su liberación, para su aceptación y luego en producción.
   d. Todas las anteriores.

---

15) Un sistema se considera modular si
   a. El sistema consiste de componentes.
   b. a) y estos componentes se pueden implementar de forma separada.
   c. b) y el cambio en un componente tiene mínimo impacto en otros componentes.
   d. c) y existe baja cohesión en estas componentes.

---

16) El mantenimiento de software se caracteriza por:
   a. Ser una actividad a la que por lo general las organizaciones de desarrollo de software le asignan la máxima prioridad y los mejores recursos si se le compara con el desarrollo.
   b. El conflicto entre las necesidades de corto plazo (solución rápida de los problemas) y las necesidades de largo plazo (mantener o mejorar los atributos de calidad del producto bajo mantenimiento y reducir los costos de mantenimiento).
   c. Representar normalmente un costo significativamente menor al costo de desarrollo.
   d. Lo fácil que normalmente resulta, inclusive para personal que no conoce el producto a modificar, estimar el esfuerzo de un cambio propuesto a partir del análisis de la documentación de diseño.

---

17) Los conceptos claves para un buen diseño son que
   a. Sea correcto.
   b. a) y cumpla el Principio abierto-cerrado.
   c. b) y exista alta cohesión y bajo acoplamiento en sus componentes.
   d. b) y exista baja cohesión y bajo acoplamiento entre sus componentes.

---

18) Seleccione el conjunto de casos de prueba con menor cardinalidad que cumple con la técnica de Particiones en Clases de Equivalencia para un método que recibe como parámetro la edad de una persona, (entero positivo) y devuelve "niño" si es menor a 12 años, "adolescente" si está entre 12 y 17 años, "joven" si está entre 18 y 32 años, "adulto" si está entre 33 y 60 y "adulto mayor" si es mayor a 60.
   a) {12,17,18,32,60}
   b) {-1,0,1,11,12,13,17,18,19,32,33,60,61, MAX-INT}
   c) {-1,0,11,12, 17,18,32,33,60,61,MAX-INT}
   d) {11,12,17,18,32,33,60,61}

---

19) La característica del software confiabilidad se define, según el modelo ISO/IEC 9126 como:
   a. Un conjunto de atributos relacionados con la capacidad del software para mantener su nivel de desempeño en determinadas condiciones por un período de tiempo.
   b. Un conjunto de atributos concerniente a la relación entre el nivel de desempeño del software y la cantidad de recursos consumidos.
   c. Un conjunto de atributos relacionados con el esfuerzo de uso para un grupo de usuarios.
   d. Un conjunto de atributos relacionados con la existencia de un conjunto de funciones y sus propiedades específicas.

---

20) La Configuración del Software:
   a. Comprende los elementos que componen toda la información generada durante del proceso de ingeniería de software.
   b. a) y algunos ejemplos pueden ser: especificación de requerimientos, conjuntos de casos de prueba, reportes de defectos, manuales de usuarios.
   c. b) pero esto no quiere decir que desde que se crea un elemento de configuración es necesario controlar los cambios mediante procedimientos formales.
   d. b) y un elemento de configuración se convierte en línea base solamente cuando lo valida el cliente.

21) La estructura de las áreas de proceso del modelo CMMI:
   a. Varía para las distintas categorías de áreas de proceso.
   b. Propone metas genéricas y metas específicas para cada área de proceso.
   c. b) y estas metas son requeridas a diferencia de las prácticas que son esperadas.
   d. Ninguna de las anteriores.

---

22) Respecto al mantenimiento:
   a. Un Comité de Control de Cambios es responsable por controlar los cambios, tanto mejoras como corrección de defectos; este comité debe conjugar las visiones de desarrolladores, usuarios y cliente para calificar (defecto o mejora), autorizar y asignar prioridad a los cambios solicitados.
   b. En la definición de procedimientos de cambios conviene detallar los pasos que se deben cumplir (evaluación, priorización, análisis de impacto, implementación, pruebas, puesta en producción) para cambios normales, no así para cambios de emergencia. En caso de que aparezca un cambio de emergencia conviene, a partir del análisis del caso concreto, definir en ese momento cómo es la mejor forma de atacarlo, posiblemente saltando pasos del procedimiento normal.
   c. Una herramienta para la Gestión de la Configuración del Software resulta esencial durante el desarrollo de software, pero no tanto durante la fase de mantenimiento.
   d. En general la probabilidad de introducir defectos durante el mantenimiento es muy baja, por lo que no vale la pena llevar a cabo pruebas de regresión antes de poner en explotación los cambios.

---

23)
   a. Datos de la industria indican que los costos de mantenimiento de los sistemas de software tienden a bajar a medida que cada sistema es sometido a intervenciones de mantenimiento debido a la sistemática mejora en su calidad.
   b. Una estrategia para rejuvenecer software (devolverle los atributos de calidad originales) es la llamada ingeniería directa que consiste en llevar a cabo análisis, diseño, construcción, prueba y puesta en explotación.
   c. Los sistemas legados (legacy systems) se caracterizan por haber sido implementados hace muchos años, con tecnologías a menudo hoy en día obsoletas; suelen ser críticos para las organizaciones, por lo que a estas les resulta particularmente complicada su sustitución y esto mismo explica su permanencia a pesar de los costos y dificultades para su mantenimiento.
   d. Medidas de la complejidad del software tales como el número ciclomático, estructuras de datos, tamaño de procedimientos y módulos en líneas de código resultan muy poco útiles para predecir el esfuerzo de mantenimiento.

---

24) Para definir un proceso es necesario identificar:
   a. Sus entradas, su lista de actividades y sus salidas.
   b. a) y la forma de medirlo.
   c. b) y los roles involucrados.
   d. c) y su desempeño y capacidad.

---

25) El proceso de verificación se compone de las siguientes fases:
   a) Prueba Unitaria, Prueba de Integración, Pruebas Alfa, Pruebas Beta, Prueba de Instalación
   b) Prueba Unitaria, Prueba de Interfaces, Prueba de Integración, Prueba de Desempeño, Prueba de Aceptación.
   c) Prueba Unitaria, Prueba de Integración, Prueba Funcional, Prueba de Instalación.
   d) Ninguna de las anteriores.

Parte 2 Ejercicios

Ejercicio 1

El siguiente método determina si un número se encuentra 2 veces o más en un ArrayList de Integer.

```
1 public bool pertenece_plus(ArrayList<Integer> lista, int num){
2  if (lista==null || lista.size()<=1){
3        return false;
4  }
5  int iterador = 0;
6  int apariciones = 0;
7  while (lista.size()>iterador){
8        if(lista.getelemen(iterador) == num){
9              apariciones++;
10             if(apariciones>=2){
11                   return true;
12             }
13       }
14  }
15  return false;
16 }
```

De acuerdo al método presentado, conteste las siguientes preguntas:

26) Cuántas trayectorias linealmente independientes identifica del código presentado?
   a) 6
   b) 4
   c) 5
   d) 3

---

27) De los siguientes conjuntos de casos de prueba ¿cuáles cumplen con el criterio de decisión/condición? (más de una opción puede ser correcta)
   a) {(null,4)}
   b) {([5,4,6,2,6,2],6);(null,4)}
   c) {([8],8),(null,6), ([5,2,4,4,5,4],4)}
   d) {([5,2,4,4,5,4],4),(null,4),([4,2],4)}

---

28) De los casos de prueba de la pregunta anterior:
   a) Alguno(s) detectan el defecto y provocan falla.
   b) Alguno(s) detectan el defecto pero no provocan falla.
   c) Los que detectan el defecto son aquellos que cumplen con el criterio decisión/condición.
   d) Todos se ejecutan satisfactoriamente y ninguno falla.

SOLUCIÓN DE PREGUNTAS PARA VERSIÓN B

26) Cuál es la mínima cantidad de casos de prueba necesarios para cumplir con el criterio de cubrimiento de caminos para el código presentado?
   a. 6
   b. 4
   c. 5
   d. 8

---

27) De los siguientes conjuntos de casos de prueba ¿cuáles cumplen con el criterio de condición múltiple? (más de una opción puede ser correcta)
   a. {([5,2,3,4,5,3],3),(null,3),([4,2],4),([1],1)}
   b. {([5,4,6,2,6,2],6);(null,4)}

c. {([8],8),(null,6),([5,2,4,4,5,4],4)}
d. {([5,2,4,4,5,4],4),(null,4),([4,2],4)}

---

28) De los conjuntos de casos de prueba de la pregunta anterior:
a. Alguno(s) detectan el defecto y provocan falla.
b. Alguno(s) detectan el defecto pero no provocan falla.
c. Los que detectan el defecto son aquellos que cumplen con el criterio de condición múltiple.
d. Todos tienen al menos un caso de prueba que falla.
