- 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 32.**
- 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) Al diseñar una interfaz de Usuario:
a. En la mayoría de los casos lo más importante a considerar son los usuarios, quiénes son, de que cultura, qué habilidades tienen, etc.
b. Un buen diseño de los mensajes de error puede ser fundamental para que el usuario se sienta cómodo con el sistema.
c. Es conveniente preocuparse por la usabilidad desde el comienzo del diseño de la interfaz de usuario.
d. Todas las anteriores.

---

2) El estilo de arquitectura
a. de programa principal y subrutinas, presenta una descomposición jerárquica basada en la relación "usa", donde la correctitud de una subrutina depende de la correctitud de las subrutinas que llama.
b. (a) y el estilo de arquitectura de tubos y filtros presenta como una ventaja que los filtros son fácilmente reutilizables. Sin embargo, la independencia de los mismos involucra repetición de procesos de preparación, por ejemplo validaciones.
c. de pizarrón (blackboard) involucra interacción directa entre las fuentes de conocimiento que realizan los cálculos correspondientes, y almacenamiento de datos en el pizarrón y este último (el pizarrón) no incide en el orden de operación definido para las fuentes de conocimiento.
d. en capas se caracteriza porque cada capa provee un conjunto de servicios a capas inferiores en la jerarquía definida, y requiere servicios brindados por capas superiores, lo que facilita el mantenimiento.

---

3) En el proceso genérico de diseño visto en el curso
a. la definición de la Arquitectura de Software se puede realizar en cualquier momento del diseño ya que lo más importante es dejarla documentada.
b. la definición de la Arquitectura de Software debe realizarse como primer paso del diseño ya que es la base sobre la que realizar el resto del diseño.
c. (b) y existen estilos de arquitectura que sirven para definirla.
d. (c) y es mejor definirla después de tener diseñados en detalle los subsistemas para asegurarse que se tengan todos los subsistemas necesarios.

---

4) Entre las características de un buen diseño se encuentra la independencia de los elementos de software definidos donde
a. la medida de esta independencia está dada por la cohesión y acoplamiento entre módulos, donde lo que se busca obtener es baja cohesión y alto acoplamiento.
b. la cohesión de un módulo refiere a cuan conectado está el módulo con el resto de los módulos definidos.
c. la cohesión de un módulo refiere a cuan focalizado está el módulo en resolver determinado aspecto del problema.
d. (c) y cuanto mayor es la independencia se facilitan aspectos como la comprensión y el mantenimiento del diseño.

---

5) La Arquitectura de Software de un sistema
a. son los subsistemas que componen el sistema, las interfaces de estos y las reglas de interacción entre ellos.
b. (a) y una de las ventajas de documentar la Arquitectura de Software es que se puede mejorar y aumentar la comunicación con los distintos stakeholders.
c. (b) y el estilo y estructura elegidos para dicha arquitectura depende más de los requerimientos funcionales que de los no funcionales.
d. Los requerimientos funcionales de un sistema no son contradictorios entre sí, por lo que siempre se satisfacen todos con la arquitectura de software escogida.

---

6) Es correcto afirmar que:
a. Realizar una Prueba (test) consiste en ejecutar un programa con el fin de provocar fallas.
b. (a) y es necesario elegir un subconjunto de las entradas del programa para testear.
c. (b) y es imposible realizar pruebas exhaustivas y probar todas las posibles secuencias de ejecución.
d. Una de las técnicas de verificación dinámica es la Inspección.

---

7) Respecto a la verificación de software.
a. La verificación unitaria es muy costosa en relación a su beneficio. Generalmente el nivel más bajo de verificación que se realiza en la industria es el de integración.
b. (a) Y en la verificación de integración, además de encontrar defectos en las unidades, se encuentran defectos en las interfaces de comunicación.

c. Es bueno combinar técnicas de caja blanca, como por ejemplo criterio de decisión, y caja negra para la verificación a nivel de sistema.
d. La verificación a nivel de sistema puede ser realizada por algunos usuarios del mismo.

---

8)
a. En las pruebas de caja blanca se crean los casos de prueba antes de la implementación del módulo a probar.
b. En las pruebas de caja blanca 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. (b) y En el momento de probar un sistema es necesario elegir un subconjunto de las entradas del programa ya que normalmente es imposible realizar pruebas exhaustivas
d. (c) y si elegimos correctamente este conjunto de pruebas, se puede demostrar (al ejecutar el conjunto de pruebas y que ninguno de los casos de prueba falle) la ausencia de fallas en el sistema

---

9) Las pruebas de regresión
a. Son pruebas concebidas específicamente para detectar fallas en el rendimiento del sistema.
b. Ayudan a detectar faltas introducidas al realizar modificaciones al código, chequeando que funcionalidades que estaban funcionando correctamente lo sigan haciendo.
c. (b) y es conveniente tener estas pruebas automatizadas.
d. (c) y solamente se pueden tener a nivel unitario.

---

10) Dadas las siguientes afirmaciones indique cuál respuesta es correcta.
1. Descubrir defectos es uno de los objetivos de la verificación.
2. Evaluar la calidad de los productos es uno de los objetivos de la verificación.
3. La corrección de los defectos detectados es parte de la verificación.
4. Es habitual en la industria de software que las pruebas unitarias las realice un equipo especializado de testing y no el equipo de desarrollo.
5. El conocimiento de las interfaces y la forma en la que interactúan las componentes de un sistema es importante para realizar las pruebas de integración.

a. Se cumple 1 pero no se cumple 2.
b. Se cumplen 1, 2 y 4 pero no se cumple 5.
c. Se cumplen 1, 2, 4 y 5 pero no se cumple 3.
d. Se cumplen 1, 2 y 5 pero no se cumplen 3 y 4.

---

11) Durante la prueba de integración se pretende detectar
a. defectos en las interfaces entre las unidades.
b. (a) y una de las estrategias de pruebas de integración es la estrategia big-bang.
c. (b) y en dicha estrategia es costoso conocer dónde están los defectos detectados en las pruebas ya que se integran todos los módulos a la vez.
d. defectos a nivel de sistema que son difíciles de detectar a nivel unitario.

---

12) Las técnicas estáticas
a. se pueden utilizar para distintos productos de software tales como requerimientos, diseño, código fuente, casos de prueba.
b. (a) y una técnica estática conocida son las recorridas.
c. buscan ejecutar el producto de software para corregir fallas.
d. son infalibles cuando se utilizan durante la prueba de sistema.

---

13) En cuanto a la Verificación:
a. Las técnicas englobadas dentro del análisis de código son técnicas estáticas y buscan encontrar defectos en el código mediante su revisión.
b. (a) y las recorridas y las inspecciones están dentro del análisis de código y permiten unificar el estilo de programación y comprobar que se cumpla con el estándar de programación, en caso de que éste último exista.
c. (b) y además, si se analiza todo el código de un programa podemos decir que se realizó una verificación formal.
d. Ninguna de las anteriores.

---

14) Conviene comenzar la planificación de la liberación de un sistema:
a. tan pronto como sea posible, para poder definir en etapas tempranas el alcance del proyecto.
b. una vez que el software ya está construido ya que si se comenzara antes el riesgo de tener que rehacer los planes es muy grande.
c. una vez que el software ya está diseñando ya que esto permite reducir los riesgos de cambios y es lo suficientemente temprano como para evitar que la duración de las tareas de preparación de la liberación afecte la fecha prevista de liberación.
d. una vez que los requerimientos del software están definidos, lo que define el alcance del producto.

15) Para la liberación del software hay que considerar:
a. la documentación y entrenamiento de usuarios.
b. (a), y de los equipos de mantenimiento y soporte.
c. (b), y la disponibilidad de capacidad para dar solución a los problemas que aparecen al inicio de la explotación de un sistema.
d. (c), y en el caso de sistemas nuevos, que no sustituyen a uno existente, es necesario prever tareas de conversión del sistema anterior al nuevo.

16) Respecto a las estrategias de liberación:
a. En general la mejor estrategia es la Big-Bang cuyo único posible inconveniente, en aquellos casos en que se sustituye un sistema pre-existente por otro nuevo, es la necesidad de dar soporte a la convivencia entre el sistema nuevo y el anterior.
b. En general la mejor estrategia es la paulatina, salvo para aquellos casos en los que el sistema es muy crítico, complejo, con muchos usuarios y múltiples ubicaciones, en los que es preferible adoptar una estrategia Big-Bang
c. Para un sistema utilizado por pocos usuarios en una única ubicación, la estrategia Big-Bang resulta en general adecuada.
d. (c), para un sistema utilizado por muchos usuarios en múltiples ubicaciones, conviene evaluar la estrategia de liberación paulatina, como una forma de manejar los riesgos asociados a la liberación.

17) El mantenimiento de software:
a. Es necesario, de forma análoga al mantenimiento de equipos mecánicos, para sustituir componentes que se desgastan a consecuencia de su uso.
b. Es necesario ya que el software que no se mantiene tiende a tornarse menos útil porque no resulta posible adaptarlo a cambios del ambiente, por ejemplo cambios en las necesidades (funcionalidad, interfaz de usuario, capacidad de procesamiento) o en la tecnología.
c. Tiene un costo que representa una porción muy baja del costo total del software.
d. Es una actividad que requiere una capacidad técnica mucho menor que la que requiere el desarrollo de software y que tiene poco impacto sobre la calidad del software.

18) Respecto al mantenimiento de software:
a. El mantenimiento correctivo refiere a la corrección de defectos presentes en el software.
b. (a), el mantenimiento adaptativo refiere a la adaptación del software a cambios en el ambiente.
c. (b), el mantenimiento preventivo refiere a mantenimiento que tiene por objeto prevenir la aparición de fallas.
d. (c), y un ejemplo de mantenimiento preventivo consiste en rescribir un componente que presenta un historial de defectos y fallas muy importante.

19) Durante el mantenimiento del software:
a. La calidad del software puede mejorar si el mantenimiento se lleva a cabo cuidando la calidad de las intervenciones además de la rapidez de su ejecución.
b. La calidad del software en general mejora como resultado de la corrección de los defectos a medida que estos aparecen.
c. La calidad del software en general empeora ya que cada intervención presenta cierta probabilidad de introducir nuevos defectos.
d. Resulta en general sencillo preservar atributos de calidad no funcionales tales como las facilidades de adaptación, de mantenimiento, de comprensión y la modularidad.

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. a) y desde que se crea un elemento de configuración es necesario controlar los cambios mediante procedimientos formales.
d. c) y un elemento de configuración se convierte en línea base cuando se ha revisado formalmente y que se ha llegado a un acuerdo.

21) Sobre CMMI:
a. Es un modelo de mejora de procesos que recomienda un conjunto de buenas prácticas.
b. a) y que presenta dos representaciones, escalonada y continua.
c. b) y Cada representación plantea alternativas para la mejora de procesos de desarrollo y mantenimiento de software.
d. c) y El modelo divide sus áreas de proceso en cuatro categorías: Ingeniería, Gestión de procesos, Gestión de proyectos y Desarrollo de las personas.

22) Dentro de las características que recomienda considerar el estándar de calidad ISO/IEC 9126 se encuentran:

a. Eficiencia: Un conjunto de atributos concerniente a la relación entre el nivel de desempeño (performance) del software y la cantidad de recursos consumidos, bajo determinadas condiciones
b. (a) y Funcionalidad: Un conjunto de atributos relacionados con la existencia de un conjunto de funciones y sus propiedades específicas.
c. (b) y Mantenibilidad: Un conjunto de atributos relacionados con la capacidad del software para ser transferido de un ambiente a otro.
d. Ninguna de las anteriores.

23) Además de la evaluación de características existen otras formas de evaluación como por ejemplo:
a. Evaluación de recursos humanos.
b. Estudios retrospectivos.
c. b) y Estudio de casos.
d. Ninguna de las anteriores

24) Se denomina Análisis Postmortem a:
a. La evaluación de un proyecto una vez terminado.
b. a) y se usa para determinar el desempeño de las personas en el proyecto.
c. b) y para mejorar las predicciones.
d. ninguna de las anteriores.

25) La madurez de la organización es la capacidad del conjunto de sus procesos. Que una organización sea madura quiere decir que:
a. la organización obtuvo una certificación de calidad de sus procesos de desarrollo de software.
b. los procesos fueron ajustados luego de ser utilizados un período de 2 años a partir de su puesta en vigencia.
c. los resultados obtenidos con sus procesos tienen una alta probabilidad de coincidir con el desempeño estimado.
d. el desempeño de los procesos es siempre el mismo.

Parte 2 Ejercicios

Ejercicio 1

El siguiente programa calcula el máximo común divisor entre dos números.

```
MCD(numero1, numero2) {
    mayor = max (numero1, numero2);
    menor = min (numero1, numero2);
    encontrado = falso;
    mientras ( ! encontrado ) {
        resto = mayor % menor; // módulo
        if (resto == 0) {
            encontrado = true;
        } else {
            cociente = mayor / menor; // división entera
            mayor = menor;
            menor = resto;
        }
    }
    return menor;
}


max (numero1, numero2){

    if (numero1 > numero2){
        return numero1;
    } else {
        return numero2;
    }
}


min (numero1, numero2){

    if (numero1 < numero2){
        return numero1;
    } else {
        return numero2;
    }
}
```

26) Indique la complejidad ciclomática del método MCD (no incluya los métodos max y min).

a. 1.
b. 2.
c. 3.
d. 4.
e. 5.

---

27) Cuántos casos de prueba se necesitan para cumplir con el criterio de trayectorias linealmente independientes para el método MCD (sin incluir los métodos max y min).

a. 2.
b. 3.
c. 4.
d. 5.
e. 6.

---

28) Considerando los siguientes datos de prueba para el método MCD Indique el conjunto con menor cardinalidad que cumple con el criterio de decisión sin tener en cuenta los tres métodos.

a. { (num1 = 2, num2 = 2) }
b. { (num1 = 5, num2 = 2) }
c. { (num1 = 5, num2 = 2) , (num1 = 2, num2 = 5) }
d. Ninguno de los conjuntos cumple con lo pedido.

---

29) Cuando un método invoca a otro se puede extender el grafo de flujo de control (GFC) del método que invoca de forma de considerar el GFC del método invocado. La técnica sencillamente agrega el CFG del invocado en el CFG del invocante en el lugar que dicha invocación se produce. Considerando la extensión del GFC del método MCD con los GFC de max y min indique la complejidad ciclomática de ese grafo extendido.

a. 2.
b. 3.
c. 4.
d. 5.
e. 6.

---

30) Considerando los siguientes datos de prueba para el método MCD. Indique el conjunto con menor cardinalidad que cumple con el criterio de decisión teniendo en cuenta los tres métodos.

a. { (num1 = 2, num2 = 2) }
b. { (num1 = 5, num2 = 2) }
c. { (num1 = 5, num2 = 2) , (num1 = 2, num2 = 5) }
d. Ninguno de los conjuntos cumple con lo pedido.

---

31) ¿Existe código innecesario que puede ser detectado mediante revisión de código?

a. Sí.
b. No.

---

32) Dado este otro programa:

```
otroMetodo(x, z)
If x <> 0 then y := 5 ;
          else z := z – x;
If z > 1 then z := z / x ;
         else z = 0;
```

a. Cualquier conjunto de datos de prueba que cumpla con el criterio de cubrimiento de trayectorias independientes, detecta la división entre cero que se puede dar al ejecutar el código.
b. Ningún conjunto de datos de prueba que cumpla con el criterio de cubrimiento de condición múltiple, detecta la división entre cero que se puede dar al ejecutar el código.
c. Existe al menos un conjunto de datos de prueba que cumple con el criterio de cubrimiento de sentencias y que detecta la división entre cero que se puede dar al ejecutar el código.
d. (c) y el conjunto de datos de prueba { (x=0 , z=5) , (x=3, z=5) } también detecta la división entre cero.
e. (d) y dicho conjunto cumple con el criterio de cubrimiento de sentencias.

Ejercicio 2

Una cadena de 13 farmacias desea comenzar a comercializar sus productos a través de Internet. La funcionalidad de la aplicación consiste en:

1. Una página Web con facilidades para consultar la lista de 18000 artículos. La consulta brinda la posibilidad de ver una imagen del artículo, su descripción y precio.
   1.1. Para ubicar un artículo se puede buscar por una categorización jerárquica.
   1.2. Se puede buscar por nombre.
2. El cliente para comprar debe registrarse (nombre, dirección, teléfono, e-mail, Cédula de Identidad, contraseña).
3. Un carro de compras se puede ir llenando/descargando a medida que se revisan los artículos.
4. El cliente puede revisar el contenido de su carro y consultar el importe que lleva gastado.
5. El cliente paga con tarjeta de crédito, para la que el sistema solicita autorización en el momento.
6. El servidor es seguro (los datos viajan encriptados).
7. Cada pedido aceptado queda identificado por un número correlativo y se emite en expedición en dos copias.
8. De los pedidos aceptados pendientes de preparar se emite un listado ordenado por ubicación del artículo en el depósito, con los datos: ubicación, artículo, código, cantidad.

Marque de la lista siguiente todas las cuestiones relevantes respecto a la Factibilidad Operativa:

a. La cantidad y calidad de recursos técnicos necesarios para implementar el sistema.
b. La cantidad de recursos financieros necesarios para que el sistema funcione día a día.
c. La cantidad de clientes que efectivamente estén dispuestos a utilizar el sistema.
d. La confianza que los clientes tengan para realizar compras por Internet.
e. Lo fácil o complicado del uso del sistema por parte de los usuarios.
f. Los recursos técnicos con los que deben contar los usuarios.
g. Los recursos técnicos y humanos necesarios para implementar el software.
h. Los tiempos de respuesta que obtengan los usuarios al usar el sistema.
i. Los plazos de entrega que efectivamente ofrezca el servicio.
j. Los tiempos e infraestructura necesarios para preparar y entregar los pedidos.
k. El porcentaje de pedidos que el servicio no pueda satisfacer.
l. El nivel de disponibilidad del servicio.
m. El costo previsto de la implementación del sistema.
n. Los horarios en los que se va a ofrecer el servicio.

Marque de la lista siguiente todos los aspectos que debiera considerar la prueba del sistema:

a. Facilidad de mantenimiento del software.
b. Facilidad de uso del software.
c. Pruebas unitarias de cada componente.
d. Evaluación de la seguridad en la compra por Internet.
e. Prueba de estrés para determinar el punto de quiebre.
f. Prueba de carga para determinar tiempos de respuesta.
g. Facilidad de instalación del software.
h. Capacidad de recuperación del sistema.
i. Prueba de la funcionalidad cubriendo ciclos completos.

Marque de la lista siguiente todos los aspectos que debiera contemplar un plan de liberación del sistema considerando que existe un único depósito de expedición con 4 empleados:

a. Entrenamiento a los usuarios finales.
b. Preparación de documentación para los usuarios, considerando la posibilidad de ayuda en línea.
c. Preparación de documentación para el personal de soporte para la solución de problemas.
d. Conversión de datos del sistema anterior.
e. Entrenamiento al personal del depósito.
f. Entrenamiento al personal de soporte para la solución de problemas.
g. Asignación de personal para las tareas de soporte.
h. Carga inicial de los datos que previamente no están disponibles en el sistema (por ejemplo las imágenes de los artículos).
i. Preparación de la convivencia con el sistema anterior.

Marque de la lista siguiente todos los aspectos de la estrategia de liberación que le parece resultan más adecuados para manejar los principales riesgos:

a. Implantar el sistema en las 13 farmacias a la vez.
b. Implantar el sistema en una farmacia e ir incorporando otras farmacias de forma paulatina.
c. Poner en funcionamiento el sistema coincidiendo con una gran campaña publicitaria en la que se promociona el uso del nuevo sistema.
d. Realizar un procesamiento en paralelo entre el sistema actual y el nuevo.
e. Poner en funcionamiento el sistema inicialmente para un conjunto reducido de clientes.