- Cada pregunta de la parte múltiple opción contestada correctamente tiene un valor de 3 puntos. Cada pregunta incorrecta de la múltiple opción resta 1 punto.
- Las preguntas 21 a 25, de la parte de ejercicios, que son contestadas correctamente valen 4 puntos cada una. Cada pregunta incorrecta resta 0,5 puntos.
- El ejercicio 26 tiene un valor de 20 puntos.

Parte 1 Múltiple Opción

1) Los atributos de calidad más relevantes para un producto de software que controla la inyección electrónica de combustible en el motor de un automóvil son:

a. adecuación al uso, estabilidad, tiempo de respuesta
b. amigabilidad de la interfaz de usuario, seguridad de la información, Adecuación al uso
c. amigabilidad de la interfaz de usuario, mantenibilidad, portabilidad
d. seguridad de la información (security), mantenibilidad, tiempo de respuesta

---

2) Respecto a los modelos de proceso:

a. el modelo en cascada tiene como principal ventaja que facilita la evaluación temprana por parte del usuario/cliente de los atributos de calidad externos del producto
b. el modelo de proceso en fases con liberaciones parciales en explotación tiene como ventaja importante que permite reducir el plazo de salida al mercado (time to market)
c. (b) a la vez que, comparado con un proceso en cascada, permite contar con una evaluación más temprana de parte del usuario de los atributos de calidad externos
d. (c) mientras que el modelo de proceso en fases con liberaciones parciales internas (no en explotación) tan solo permite que el usuario evalúe los atributos de calidad internos.

---

3) Dado un archivo con información de un empleado (número, nombre, cargo, y CI y nombre del cónyuge de los que están casados). ¿Cuántos DETs se deben contar en la impresión de toda la información de un empleado?

a. una EQ con 3 DETs
b. una EQ con 5 DETs
c. 2 EQ, una con 3 DETs y otra con 2 DETs
d. 2 EQ, una con 3 DETs y otra con 5 DETs

---

4) Lo mejor ante un conflicto es:

a. tratar de ignorarlo
b. que cada uno ceda un poco
c. enfrentarlo y buscarle una solución
d. que la persona con autoridad tome una resolución

---

5) La administración de los requisitos

a. Se hace antes del proceso de requisitos
b. Se hace en paralelo con el proceso de requisitos
c. Se hace a continuación del proceso de requisitos
d. Comienza cuando se tiene una primera versión del documento de requisitos

---

6) Cuando se verifica el documento de requisitos, se debe revisar si es:

a. No ambiguo, es decir, que no haya contradicciones.
b. Realista, es decir que corresponda a requisitos reales del usuario y no inventados
c. Consistente, es decir que no falte nada
d. Verificable, es decir que para cada requisito existe un proceso finito para determinar que el sistema lo cumple.

---

7) Un/Los caso(s) de uso

a. es un conjunto de escenarios posibles
b. (a) donde cada escenario es una secuencia de acciones entre los actores y el sistema
c. son parte del análisis orientado a objetos
d. siempre guían el proceso de desarrollo

---

8)

a. La definición de la Arquitectura de Software se puede realizar luego del diseño detallado
b. (a) y a partir de los subsistemas definidos se realiza el diseño de cada uno de estos en detalle
c. La definición de la Arquitectura de Software debe realizarse como primer paso del diseño del sistema
d. (c) y a partir de los subsistemas definidos se realiza el diseño de cada uno de estos en detalle

---

9) El proceso de diseño tiene distintos niveles de abstracción, siendo el de definición de la Arquitectura
a. de las últimas actividades en realizarse, ya que en etapas tempranas del diseño difícilmente se cuenta con el conocimiento necesario para poder definirla en forma correcta.
b. en el que se deben concentrar los esfuerzos del equipo de diseño para identificar una correcta modularización del sistema que cumpla los requerimientos establecidos.
c. (b) y es importante tener en cuenta las características (ventajas y desventajas) de los estilos de Arquitectura existentes para poder evaluar su adecuación a la aplicación en desarrollo.
d. (c) y a esa altura del proceso, no es importante aún la definición de interfaces entre módulos, ya que posiblemente los cambios que todavía sufra la aplicación obliguen a re-definirlas varias veces.

10) El estilo tubos y filtros
a. Descompone el sistema en módulos funcionales (filtros)
b. (a) y donde a cada filtro le llegan datos, los transforman y los pasan a través de un tubo
c. (b). Este estilo tiene como ventaja que los filtros son reutilizables
d. Ninguna de las anteriores

11) Cuál es la ventaja del *dynamic binding* en SOA (Arquitectura Orientada a Servicios):
a. Permite reuso de sistemas enteros en nuevas áreas de aplicación
b. Permite contactarse con distintos proveedores de servicios en tiempo de ejecución según, por ejemplo, la disponibilidad de los servicios.
c. Permite obtener una arquitectura de software más segura
d. Mediante el uso de dynamic binding es fácil implementar la arquitectura con un lenguaje orientado a objetos

12) En las pruebas de caja blanca
e. se crean los casos de prueba antes de la implementación del módulo a probar.
f. se obtienen los casos de prueba únicamente a partir del código fuente.
g. 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.
h. Ninguna de las anteriores

13) Las pruebas de regresión
a. ayudan a detectar faltas introducidas al realizar modificaciones al código, chequeando que funcionalidades que estaban funcionando correctamente lo sigan haciendo
b. (a) y no conviene tener estas pruebas automatizadas porque al usarse luego de un cambio en el código es lógico que estás pruebas también cambien.
c. (a) y es conveniente tener estas pruebas automatizadas.
d. (c) y pueden ser tanto unitarias, de integración y de sistema.

14) Dadas las siguientes afirmaciones indique cuál respuesta es correcta
I) Descubrir defectos es uno de los objetivos de la verificación
II) Evaluar la calidad de los productos es uno de los objetivos de la verificación
III) La corrección de los defectos detectados es parte de la verificación
IV) Es normal en la industria de software que las pruebas unitarias las realice un equipo especializado de testing y no el equipo de desarrollo
V) 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 I pero no se cumple II
b. Se cumplen todas
c. Se cumple I, II, IV y V
d. Se cumplen I, II y V

15) En las pruebas de caja negra
a. No se necesita disponer del código fuente para generar los casos de prueba
b. (a) y se generan los casos de prueba a partir de la especificación
c. (b) y porciones enteras de código pueden quedar sin ejecutar al correr los casos seleccionados con esta técnica
d. (c) y una técnica de caja negra es “Partición en clases de equivalencia”

16) Con respecto a la liberación de un producto
a. conviene prever actividades de soporte a llevar a cabo durante el inicio de la explotación, período en el que normalmente se producen problemas, los que pueden estar originados en diversas fuentes: software, capacitación de los usuarios, documentación de usuarios, procedimientos, conversión de datos, software de base, hardware, entre otros
b. (a) y una forma de reducir los riesgos que este cúmulo de problemas puede traer en casos en los que la envergadura del software a instalar involucre muchos usuarios o múltiples ubicaciones, consiste en una implantación paulatina
c. (b) y el comienzo de la planificación de las actividades de soporte al inicio de la explotación conviene que sea una vez completada la capacitación a los usuarios sobre el software a implantar
d. la mejor estrategia de implantación es alguna variante de implantación paulatina, ya que la estrategia big-bang tan solo tiene desventajas

17) En el mantenimiento
a. El análisis de impacto de un cambio a menudo es de las actividades que requieren un mayor esfuerzo.
b. Ante distintas alternativas de solución conviene implementar aquella que permita implementar los cambios en el menor tiempo posible
c. Ante distintas alternativas de solución conviene implementar aquella que asegure preservar los atributos de calidad del producto a largo plazo
d. En general, una buena política para asegurar un uso eficiente de los recursos dedicados a mantenimiento, consiste en liberar cada cambio o corrección lo más pronto posible.

18) El estándar de calidad ISO/IEC 9126 es utilizado para:
a. Evaluar productos de software
b. a) y Definir los requerimientos de calidad de un producto de software
c. Evaluar el proceso de desarrollo de software
d. ninguna de las anteriores

19) En la categoría Ingeniería de áreas de proceso del modelo CMMI se incluye:
a. La gestión de requerimientos
b. (a) y la gestión de riesgos
c. (a) y la solución técnica, la verificación y validación
d. (c) y el desarrollo de los requerimientos y la integración del producto

20) El análisis post-mortem de un proyecto no se recomienda:
a. Cuando se constata que el proyecto ha sido todo un éxito
b. Cuando se constata que el proyecto ha fracasado
c. Cuando todavía no han pasado seis meses de culminado el proyecto
d. Ninguna de las anteriores

Parte 2 Ejercicios

21) Dado el siguiente programa (que supondremos correcto):

```
read(x); read(y);
if x = y
    then  z := 2;
    else   z:= 0;
endif
write z;
```

y el siguiente conjunto de datos de prueba: { (x = 3, y = 2), (x = 3, y = 1), (x = 3, y = 3) }
Supongamos ahora que el programador se equivocó al programar y en lugar de
then z := 2 escribió
then z := 22

Al ejecutar el conjunto de casos de prueba:

a. El programa a veces falla y a veces no
b. El programa no falla nunca
c. El programa falla siempre
d. ninguna de las anteriores

---

22) Un método "mult2" de una clase, recibe un entero >= 1 y devuelve un booleano cuyo valor es true si el entero pasado es múltiplo de 2 y false en caso contrario.

bool mult2 (int a)

Se considera que el método está definido por contrato por lo que no debe recibir números menores que uno y tampoco el método lo debe controlar.Se planifica realizar pruebas de caja negra de este método usando las técnicas de partición en clases de equivalencia y valores límites. Observando la salida se puede partir la entrada en dos clases de equivalencia, aquellos valores que son múltiplo de dos y los que no. Mirando la entrada parece interesante probar con el valor límite 1 y con el valor límite MaxInt (máximo valor entero que acepta el lenguaje de programación usado y que es múltiplo de dos). Considerando nuevamente la salida parece interesante probar con el valor más chico que sea múltiplo de dos, el cual es 2 y con el valor más grande que no sea múltiplo de dos el cual es (MaxInt -1). Además se toma un valor cualquiera (sin ser uno ya tomado) de cada una de las dos clases de equivalencia. Por lo tanto nuestro conjunto de datos de prueba es el siguiente: {1, 2, MaxInt - 1, MaxInt, x, y} siendo x un valor múltiplo de 2 y distinto de 2 y de MaxInt y siendo y un valor que no es múltiplo de 2 y distinto de 1 y MaxInt -1.

¿Cuál de las siguientes opciones es correcta?

a. El conjunto de datos de prueba cumple con el criterio de cobertura de sentencias.
b. a) y también con el de decisión
c. b) y también con el de condición múltiple
d. ninguna de las anteriores.

---

23) Dado el método explosionLista indique el conjunto de datos de prueba con menor cardinalidad y que cumple con el criterio de decisión de las opciones listadas a continuación.

```
void explosionLista (lista : Lista) {
   int a = 0;
   while lista.tieneMasElementos() and a < 2{
      lista.borrarPrimerElemento();  //reduce la cant. de elem. en la lista en 1
      a = a+1;
    }
}
```

a. CDP = { lista = vacía }
b. CDP = { lista = vacía ; lista = {el1, el2, el3, el4} }
c. CDP = { lista = vacía ; lista = {el1, el2} ; lista = {el1, el2, el3, el4, el5, el6} ; lista = {el1} }
d. CDP = { lista = {el1, el2} ; lista = {el1, el2, el3, el4, el5, el6} ; lista = {el1} }

---

24)

```
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) } detecta la división entre cero.

---

25) Dada una especificación de un programa, el código del mismo y casos de prueba:

Especificación:
El programa recibe un número natural como entrada y devuelve como resultado un número primo. Dicho número primo debe ser estrictamente mayor al número recibido como parámetro y a su vez no debe existir otro número primo menor estricto que él y mayor estricto que el número ingresado como parámetro. Es decir, el número que se devuelve es el número primo estrictamente mayor y "más cercano" que el dado como entrada.
Más formalmente:
Pre-Condición: C : Natural
Post-Condición: J : Natural tal que J es primo and J > C and (not exists H tal que H: Natural and H es primo and H < J and H > C)

Programa

```
Program primoMayor (in : Natural C) : Natural
var J : Natural;
begin
    if (C = 0) return 1;
    if(C = 1) return 2;
    if (C = 2) return 3;
    J := 2
    while (C >= 3) do
        J := J + 3;
       C := C – 1;
    end-while
    return J;
end begin
```

Casos de prueba

	Dato	**Resultado esperado**
1)	0	1
2)	1	2
3)	2	3
4)	8	11

a. El programa no es correcto. Además, los casos 1 y 4 cumplen con el criterio de cobertura de sentencias
b. El programa no es correcto. Además, todos los casos juntos cumplen con el criterio de decisión.
c. El programa es correcto. Además, todos los casos juntos cumplen con el criterio de decisión.
d. No se puede cumplir con el criterio de decisión porque cuando el número ingresado es menor o igual que 2 no se ejecuta el "while" y cuando el número ingresado es mayor que 2 el "while" se hace siempre true y no false.

26)

En la tabla siguiente se presentan las tareas de un proyecto de desarrollo de software con la fecha de inicio planificada, su fecha de fin y sus tareas previas. Se considera que los días de inicio y fin se incluyen completamente dentro de la duración de la tarea.

Tarea	Día Inicio	Día Fin	Tareas previas
1	1	5	
2	1	3	
3	1	7	
4	7	8	
5	6	8	1, 2

Realice un diagrama de Gantt del proyecto.

Realice un grafo de actividades. Calcule la duración total del proyecto e indique las tareas que están en el camino crítico.

En la tabla siguiente se presentan las tareas con la fecha de inicio planificada, duración, la persona asignada para llevarla a cabo y un porcentaje de dedicación a la tarea.

Tarea	Día Inicio	Día Fin	Tareas previas	Responsable	Dedicación %
1	1	5		Pedro	100
2	1	3		Juan	50
3	1	7		Juan	50
4	7	8		Pedro	50
5	6	8	1, 2	Juan	100

Realice un perfil de uso de recursos para cada responsable. Indique si los recursos están sobreutilizados, subutilizados u ociosos y en qué períodos.

Indique si esta asignación es factible.

Si tuviera que reducir la duración del proyecto, indique tres opciones distintas.