1. Un diseño debe ser
   a) verificable
   b) (a) y completo respecto a los requerimientos del producto
   c) (b) y lo más simple posible
   d) Ninguna de las anteriores.

---

2. Un sistema modular permite
   a) Que los distintos módulos sean desarrollados por distintos equipos de desarrollo (o personas)
   b) (a) y que sea más fácil su verificación
   c) (b) y que se encuentren más rápidamente los defectos mientras se realizan las pruebas
   d) finalizar en plazo un proyecto

---

3. El diseño
   a) No debe seguir la arquitectura propuesta porque probablemente esta no sea correcta
   b) Lo debe realizar una única persona
   c) Sirve de base para la programación
   d) Debe ser realizado en UML

---

4. Existen micro-procesos para la fase de programación
   a) Uno muy conocido es XP
   b) Normalmente en estos procesos se indica que no se debe abandonar la fase de programación mientras se detecten (o provoquen) fallas
   c) Normalmente en estos procesos se indica que no se debe abandonar la fase de programación mientras la unidad contenga defectos
   d) Uno muy conocido es el modelo de proceso iterativo e incremental

---

5. Dado el siguiente programa y una de sus ejecuciones simbólicas asociada, decir cuál es la opción más correcta, siendo ent1 y ent2 dos variables enteras.

```
[ent1 = A, ent2 = B]
if (ent1 > ent2)
   then begin
        write ("primer then");
[ent1 = A, ent2 = B] [A > B] [se imprimió "primer then"]
        if (ent2 > ent1) then write ("segundo then");
                         else  write ("segundo else");
[ent1 = A, ent2 = B] [A > B and B > A] [se imprimió: "primer then", "segundo then"]
[ent1 = A, ent2 = B] [false] [se imprimió: "primer then"] – Esta ejecución no es posible. No
se imprime "segundo then".
        endif
    end
    else  write ("primer else");
endif
```

   a) Un "false" en una ejecución simbólica siempre indica que el camino de ejecución realizado nunca se puede ejecutar.
   b) Un "false" en una ejecución simbólica siempre indica código inalcanzable.
   c) La ejecución simbólica dada no es correcta en sí misma.
   d) Ninguna de las anteriores.

---

6. Dado el siguiente código Java indique cuántos caminos simples existen en el mismo. Un camino es simple si es un camino ejecutable y no ejecuta dos veces una misma sentencia (a menos que dicha sentencia sea una decisión {**if**, **for, while, etc.**}).

```
public String algo(String[] entrada) {
    if (entrada == null) return null;
    String concat = "";
    for (int index=0; index<entrada.length; index++){
        concat = concat + entrada[index];
    }
    if (concat.length() > 5) return "mas 5";
    return ""+concat.charAt(0);
}
```

a) 3
b) 4
c) 5
d) 6

---

7. Un programa recibe un arreglo de Strings y devuelve un String. En caso que el arreglo sea nulo devuelve nulo. En caso que los arrays concatenados formen un array de más de 5 elementos devuelve el String “mas 5”. Sino devuelve un String que sólo contiene el primer carácter del String que está en la primera posición del arreglo pasado. Se tiene el siguiente programa para dicha especificación:

```
public String algo(String[] entrada) {
    if (entrada == null) return null;
    String concat = "";
    for (int index=0; index<entrada.length; index++){
        concat = concat + entrada[index];
    }
    if (concat.length() > 5) return "mas 5";
    return ""+concat.charAt(0);
}
```

a) El caso de prueba (entrada = null, resultado esperado = null) ejecuta según lo esperado
b) (a) El caso de prueba (entrada = {“uno”, “dos”}, resultado esperado = “mas 5” ejecuta según lo esperado
c) (b) El caso de prueba (entrada = {“uno”}, resultado esperado = “u”) ejecuta según lo esperado
d) (c) Considerando los casos de prueba de a, b y c se cumple con el criterio de decisión.

---

8. Se encuentra la siguiente descripción de un sistema en la revista Pequeños Sistemitas Funcionando de la IEEE del 12 de octubre de 2004: “El sistema deberá soportar 130 usuarios conectados a la vez. El mismo deberá poder ejecutarse tanto en Linux como en Windows con Oracle o Informix como manejadores de bases de datos”. Probar el sistema para Windows con Oracle, Windows con Informix, Linux con Oracle y Linux con Informix es una prueba de:
a) estrés (esfuerzo)
b) configuración
c) volumen
d) de datos

9.
a) Las técnicas de verificación estática, en comparación con las dinámicas, son más efectivas en la detección temprana de defectos
b) Las técnicas de verificación dinámica tienen una validez general y no dependen del caso de prueba específico
c) La verificación estática no sirve para verificar el documento de especificación de requerimientos
d) Para un proyecto de desarrollo se debe definir qué tipo de técnica usar (o estática o dinámica). Esto se debe a que las mismas no son complementarias.

10.
a) La verificación es un proceso caro en comparación con otros procesos del desarrollo de software.
b) (a) y por esto es conveniente no realizar ningún tipo de verificación durante el desarrollo
c) Las pruebas a nivel de sistema se deben detener cuando no se detecten más fallas.
d) Ninguna de las anteriores.

---

11. La conversión de datos de un sistema viejo a un sistema nuevo
a) En el caso de sistemas de información es un aspecto que normalmente no requiere mayor esfuerzo.
b) Conviene considerarla al comienzo del proyecto que involucra la sustitución, para tomarla en cuenta en la definición del alcance y en la planificación general del proyecto.
c) Conviene considerarla una vez que el sistema nuevo ya está construido, para evitar re-trabajo proveniente de eventuales cambios en las estructuras de datos.
d) Conviene definirla por completo al comienzo del proyecto, por más que la estrategia de implantación no esté definida, ya que la estrategia de implantación y la conversión de datos son cuestiones independientes.

---

12. La capacitación y el entrenamiento a usuarios
a) Son tareas que se pueden planificar de forma completamente independiente de las tareas de documentación del software, ya que no guardan relación entre ellas.
b) Son tareas que dado un determinado producto a construir, inevitablemente terminan quedando comprendidas en el alcance del proyecto.
c) En proyectos en los que la población a capacitar es muy grande, puede resultar conveniente capacitar a un grupo de capacitadores, para que estos luego brinden la capacitación al resto de la población.
d) (c), y estos capacitadores pueden llegar a brindar un primer nivel de atención para la resolución de problemas, una vez que el software entre en producción.

13. El mantenimiento de software:
a) Es una tarea que requiere poco esfuerzo y que en general resulta poco importante para la economía de las organizaciones de desarrollo de software
b) (a) y un producto de sofware en general cambia, o se vuelve menos útil
c) La calidad del mantenimiento puede afectar positiva o negativamente diversos atributos de calidad del software, entre ellos su mantenibilidad, confiabilidad y adaptabilidad.
d) (c) y si el mantenimiento se realiza de forma adecuada, un producto de software podría vivir durante un tiempo muy prolongado

---

14. Respecto al mantenimiento de software:
a) 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
b) (a), en particular, para poder llevar a cabo una adecuada prueba de regresión del sistema de forma eficiente ya que esta prueba tiene costos fijos significativos que es preferible distribuir entre varios cambios
c) (b), esa política debiera estar complementada por procedimientos de emergencia que permitan atender problemas que no pueden esperar hasta la próxima liberación planificada
d) Una caída en el esfuerzo destinado al mantenimiento correctivo puede estar indicando problemas en la calidad del desarrollo o del mantenimiento

---

15. El rejuvenecimiento de software
a) Tiene por objetivo mejorar los atributos de calidad de un producto de software que se hayan degradado como resultado de tareas de mantenimiento
b) (a), en general se trata de mejorar, entre otros, la mantenibilidad y confiabilidad
c) (b), y una forma de rejuvenecimiento está dada por la reingeniería que consiste en generar documentación del diseño a partir del código fuente
d) Consiste en recompilar todos los ejecutables a partir de los fuentes para asegurar la consistencia entre fuentes y ejecutables.

---

16. El estándar de calidad ISO/IEC 9126 es utilizado para:
    a) la evaluación de productos de software
    b) la evaluación del proceso de construcción de software
    c) las dos anteriores
    d) ninguna de las anteriores

---

17. En un proceso de evaluación de un producto no alcanza con identificar los requerimientos de calidad y seleccionar las métricas a utilizar. Además es preciso:
    a) Definir los niveles de puntuación
    b) Definir el criterio y el procedimiento de evaluación
    c) Medir, otorgar un puntaje y dictaminar
    d) Todas las anteriores

18. La relevancia de las características de calidad y los atributos asociados de un producto de software:
    a) Son homogéneas para todos los productos de software
    b) Varían según las distintas categorías de software
    c) Dependen solamente del criterio del evaluador
    d) Ninguna de las anteriores

---

19. El modelo CMMI es:
    a) un estándar de calidad para el desarrollo de software
    b) el proceso de desarrollo de software más recomendable
    c) un modelo de mejora de procesos que recomienda un conjunto de buenas prácticas para la construcción y mantenimiento de software
    d) un conjunto de actividades definidas para lograr una mejor calidad en los productos de software

---

20. El análisis post-mortem de un proyecto se recomienda:
    a) Cuando se constata que el proyecto ha sido todo un éxito
    b) Cuando se constata que el proyecto ha fracasado y se quiere identificar las causas y los responsables
    c) Cuando han pasado por lo menos seis meses que el proyecto ha culminado
    d) Siempre, al culminar un proyecto para identificar los aciertos y las dificultades con el fin de aprender para el futuro

Ejercicio – 20 puntos

Un programa recibe una lista numerada de listas de números naturales no ordenados.

Ejemplo de Entrada:

```
{[1, 2, 3], [3, 5, 3], [4, 4, 2, 3, 3], [3, 3, 3, 4, 4]}
```

Como salida el programa presenta:

- El promedio de cada elemento de la lista numerada.
- El promedio de los promedios pero eliminando los elementos con mayor y menor promedio (uno sólo en cada caso para situaciones donde el máximo y/o el mínimo lo comparten varios elementos).
- Los promedios devueltos al usuario son el redondeo natural del promedio real. Considerar que x,5 redondea a x+1.

Ejemplo de Salida:

```
Promedio 1 = 2
Promedio 2 = 4
Promedio 3 = 3
Promedio 4 = 3
Promedio general = 3
```

Se elimina el promedio1 por ser el mínimo y el promedio2 por se el máximo de los promedios.

El método Java que implementa lo presentado tiene la siguiente firma:
public Resultado promedio(ArrayList<ArrayList<Natural>> listaNumerada)
En caso de situaciones no especificadas retorna null

A continuación se presentan datos de entrada para realizar pruebas al método presentado. Cada una de las opciones desde la 21) a) hasta la 22) b) de su hoja múltiple opción corresponde a un único dato de entrada; es decir, a una **listaNumerada**.
Usted debe armar un conjunto de datos de prueba para probar el método. El conjunto se arma marcando en su hoja múltiple opción las opciones que se corresponden con los datos de entrada que quiera incluir en dicho conjunto. Cada dato de entrada seleccionado debe intentar probar comportamientos diferentes del método (considerar partición en clases de equivalencia). O, lo que es lo mismo, dos datos de entrada incluidos en el conjunto no deberían intentar probar el mismo comportamiento del método.

Pregunta 21) (contestar usando 21 y 22)
21) a) listaNumerada = new listaNumerada()
21) b) listaNumerada = **[** null, [1,2,3], [2,2], [2, 1, 2, 1] **]**
21) c) listaNumerada = **[** [2,3,4], [4,5,3] **]**
21) d) listaNumerada = **[** [2,2,4], [4,4,3] **]**
21) e) listaNumerada = **[** [], [1,1,1], [2,2,2], [3,3,3] **]**
22) a) listaNumerada = **[** [2,2,4], [4,4,3], [4,3,2,1,0], [1,1,1], [1,1,1] **]**
22) b) listaNumerada = **[** [2,2,4], [4,4,3], [4,3,2,1,0], [10,10,10], [10,10,10] **]**

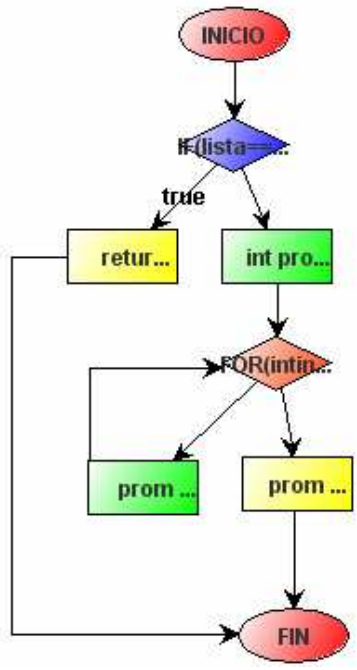
Pregunta 23)
Proponga dos datos de prueba que complementen el conjunto de datos de prueba seleccionado en la parte anterior. Estos nuevos datos deben intentar no repetir comportamientos del programa que ya fueron probados. Asuma que UNICAMENTE se pueden tener naturales en las listas de naturales.

Pregunta 24)
Ahora el programa construye la lista numerada de lista de naturales extrayendo los datos desde un manejador de bases de datos. Realizar una prueba del programa con el manejador de bases de datos caído es una prueba:
24) a) Funcional del programa
24) b) Que busca conocer la robustez del programa
24) c) De recuperación. Busca conocer como se recupera el programa luego de un fallo (manejador caído)
24) d) Unitaria de caja blanca basada en el flujo de control

Pregunta 25)
Parte del programa se ha implementado de la siguiente manera:

```
1     public Integer promedio(ArrayList<Integer> lista){
2           if (lista==null || lista.size()==0){
3                 return null;
4           }
5           int prom = 0;
6           for(int index=0; index<lista.size(); index++){
7                 prom = prom + lista.get(index).intValue();
8           }
9           prom = prom/lista.size();
10          return new Integer(prom);
11    }
```

El grafo de flujo de control del método es el siguiente (grafo generado con CFGViewer):



Un camino es simple si es un camino ejecutable y no ejecuta dos veces una misma sentencia (a menos que dicha sentencia sea una decisión {**if, for, while, etc.**}). El criterio de cubrimiento de caminos simples se cumple cuando todos los caminos simples fueron ejecutados.

Parte a) Dibujar los caminos simples del grafo de flujo de control presentado.
Parte b) Brindar un caso de prueba que ejecute el camino más corto (en cantidad de aristas) de los presentados en la parte a).
Parte c) Suponer que la sentencia 7 es ahora la siguiente:

```
prom = prom + lista.get(0).intValue();
```

Explique y **justifique brevemente** si cumplir con el criterio de cubrimiento de caminos simples asegura que el defecto provoque una falla.