1. Describa cómo evaluaría la calidad de una universidad antes de inscribirse. ¿Cuáles factores serían importantes? ¿Cuáles tendrían importancia crítica?

2. ¿Cómo definirías calidad de software?

- Proceso eficaz de software que se aplica de manera útil que proporciona valor medible a quienes lo producen y quienes lo utilizan

3. Describa con sus propias palabras el dilema de la calidad del software.

- Probar mucho o muy poco "perfect is the enemy of the good"

4. Explique la diferencia entre un error y un defecto.

- Error: Acción humana que produce un defecto (causa)

- Defecto: Imperfección o deficiencia en un producto, el cual no cumple sus requerimientos o especificaciones y necesita ser reparado o remplazado (efecto)

5. ¿Por qué no puede esperarse a las pruebas para encontrar y corregir todos los errores del software?

- Muy costoso

6. ¿Cuáles son los pasos para una revisión técnica formal?

- Preparación, examinación (autor presenta lógica, presentador lee código, reseñadores preguntan, registrador documenta, moderador mantiene enfoque, determina salidas), retrabajo, validación, salida

7. Explique la diferencia entre una revisión técnica informal y una revisión técnica formal

- Informal: Mas casual

- Formal: Preparación, roles, agenda

8. La calidad y confiabilidad son conceptos relacionados, pero difieren en lo fundamental por varias razones. Analice las diferencias.

- Confiabilidad: Capacidad del software de mantener su nivel de ejecución bajo condiciones normales en un periodo de tiempo establecido

- Calidad: Abarca la confiabilidad y mas

9. ¿Cómo están asegurando la calidad en el proyecto de equipo de esta clase? ¿Cómo la mejorarías? ¿Qué pasos están siguiendo y qué harías diferente?

10. Con sus palabras, describa la diferencia entre verificación y validación. ¿Ambas usan los métodos de diseño de casos de prueba y estrategias de pruebas?

- Validación: Garantizar que el software cumple con lo especificado originalmente

- Verificación: Actividades que garantizan que la implementación fue precisa con los requerimientos

11. ¿Por qué un módulo altamente acoplado es difícil para la prueba de unidad?

- No se puede separar

12. El concepto de "antierrores" es una forma extremadamente efectiva de brindar asistencia de depuración interna cuando se descubre un error: a) Desarrolle un conjunto de lineamientos para antierror. b) Analice las ventajas de usar la técnica. c) Analice las desventajas.

Antierror: Un buen diseño anticipa las condiciones de error y establece rutas de manejo de errores para enrutar o terminar limpiamente el procesamiento cuando ocurre un error

- Enruta o termina limpiamente un proceso cuando ocurre un error

- Ejemplos: Dividir entre 0, acceder a un índice incorrecto (overflow), etc..

- Try catch

- Assert

13. ¿Quién debe realizar la prueba de validación: el desarrollador o el usuario del software? Justifique su respuesta.

- Usuario

14. ¿Explique que es un caso de prueba? ¿Cuál es el objetivo de probar?

- Par de <entrada, salida esperada>

- Objetivo: Encontrar un error que no ha sido detectado

15. ¿Cuáles son las diferentes fuentes que tenemos para diseñar / seleccionar casos de prueba?

- Requerimientos, código, documentación, nuestras expectativas, dominio, etc…

16. ¿Cuáles son los roles en una revisión técnica formal?

- Moderador, autor, presentador, reseñadores, registrador / secretario, observadores

18. Un ingeniero de pruebas genera 50 mutantes de un programa P y 150 casos de prueba para probar dicho programa P. Después de la primera iteración de "mutation testing", el ingeniero encuentra 36 mutantes muertos y 5 mutantes equivalentes. Calcule el "mutation score" de dicha situación. ¿El ingeniero de pruebas debería crear más casos de pruebas adicionales? Justifique su respuesta

- (100*36) / (50-5) ≅ 80%

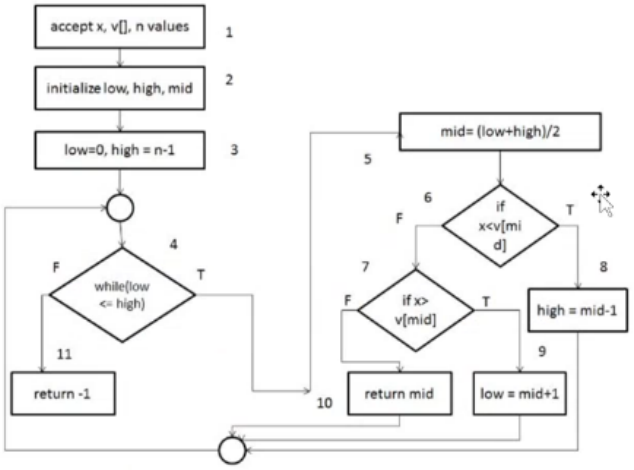
19. Existe un debate de si el código debe ser compilado (dinámico) antes de ser revisado (estático) o viceversa. En base a tu experiencia, ¿cuál es tu opinión respecto a este debate?

- Libre, con fundamentos

Dado una función en C llamada "binsearch", conteste las preguntas 20 al 23. El arreglo de entrada V[] se asume que esta ordenada de forma ascendente, n es el tamaño del arreglo y el objetivo de la función es encontrar el índice en donde se encuentra el elemento X. Si X no es encontrado, la función devuelve un "-1"

```
int binsearch(int X, int V[], int n){
   int low, high, mid;
   low = 0;
   high = n - 1;
   while (low <= high) {
      mid = (low + high)/2;
      if (X < V[mid])
           high = mid - 1;
      else if (X > V[mid])
           low = mid + 1;
      else
           return mid;
    }
    return -1;
}
```

20. Dibuja el CFG (Control Flow Graph) de binsearch()



21. Identifica paths completos para satisfacer el criterio de "Complete Statement Coverage"

22. Elige solo un path y de él deriva los "path predicate expressions"

23. Resuelve los "path predicate expressions" para generar datos de entrada.

24. Menciona cuales son las diferencias de Control Flow Testing y Data Flow Testing

- Control: Asignación, condicionales. Ejecutar las instrucciones de código correctamente
- Data: Programa acepta entrada, hace computaciones, asigna valores a variables, y produce salidas. (estático, dinámico)

25. En Data Flow Testing, explica que es una "Data Flow Anomaly" y explica si la presencia de una anomalía implica que el programa producirá resultados incorrectos.

- Forma anormal de hacer las cosas (definición, referencia, indefinición)
- No implica resultados incorrectos

26. Explica que es Domain Testing

- Dominio de entrada: Set de posibles entradas
- paths del programa: secuencia de instrucciones

27. ¿Qué es un error computacional y un error de dominio?

- Computacional: Path correcto pero resultado incorrecto
- Dominio: Path incorrecto

28. Escribe una función con un ejemplo de error de dominio

- def mayorEdad(x):
  - if x >= 18:
  - ....

29. Menciona y explica los tipos de errores de dominio que existen.

- Error de cierre: < en vez de <=
- Shifted-Boundary error: error en constante x + 4 en vez de x + 2
- Tilted-Boundary Error: Error en coeficiente: .25*x en vez de 25*x

30. Explica las siguientes definiciones: Closed boundary, Open boundary, Closed dom domain, Extreme point. Adjacent Domain, ON point, OFF point

- Closed boundary: Los puntos en el limite son incluidos en el dominio
- Open boundary: Los puntos en el limite, no son incluidos en el dominio
- Closed domain: Todos sus limites estan cerrados
- Open domain: Al menos un limite esta abierto
- Extreme point: Punto que toca 2 o mas limites
- Adjacent domain: Dominio el cual comparten un limite
- ON point: Un punto que esta en el limte, o muy cerca
- OFF point: Punto que esta lejos del limite
    - Limite abierto: A dentro del dominio
    - Limite cerrado: A fuera del dominio

31. ¿Cuál es la diferencia en hacer una prueba de caja negra y una prueba de caja blanca?

- Negra: No acceso al código
- Blanca: Acceso al código

32. Si todas las pruebas unitarias fueron exitosas, ¿Por qué es necesario hacer pruebas de integración?

- Errores de integración (interfaces, cambios, etc...)

33. Menciona y explica los tipos de prueba de integración.

- Intrasistema: Bajo nivel, combinar modulos
- Intersistema: Alto nivel, no depende de interfaces, una funcionalidad a la vez
- Por pares: Medio, un par a la vez
- Incremental, top-down, bottom-up, sandwich, Big Bang

34. Dado el diagrama de abajo, ¿Cuál sería el orden de integración por integración "Top-Down"?

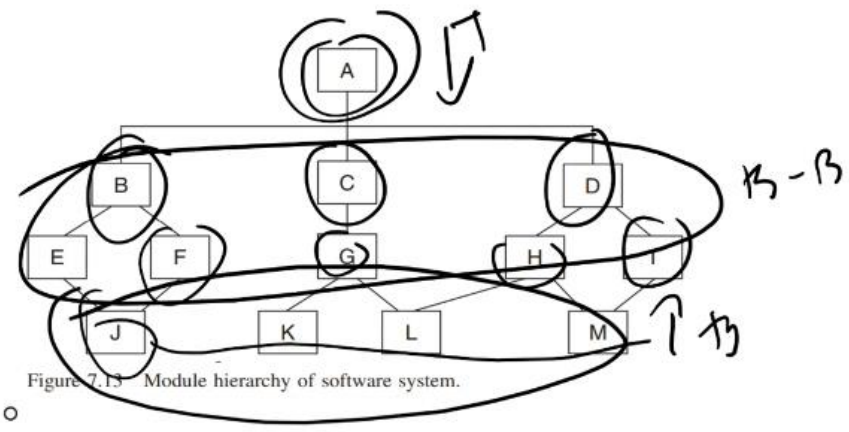


Figure 7.13 Module hierarchy of software system.