Corrección

¿Cómo defines el concepto de corrección?

El grado en que una aplicación satisface las especificaciones y consigue los objetivos encomendados por el cliente.

¿Cuando consideras que un programa es correcto?

Un programa es correcto si se realiza de acuerdo a sus especificaciones

¿Cuáles técnicas conoce para medir corrección y cómo describiría cada una de ellas?

Test: proveen evidencias convincentes respecto a que el programa hace el trabajo esperado.

Verificación: es el proceso de analizar las postcondiciones en función de las precondiciones establecidas.

Walkthrough: Recorrer el programa ante una audiencia

Debugging: es el proceso de localización del error

¿Si un programa es correcto también es eficiente?

Se define eficiencia como una metrica de calidad de los algoritmos. Asociada con una utilizacion óptima de los sistemas de computo donde se ejecutará el programa, principalmente la memoria y el tiempo de ejecución empleado.
Un programa puede tener varias soluciones correctas, sin embargo el tiempo de ejecución y memoria de cada solución puede ser muy diferente

¿Un programa que no presenta errores de compilación es correcto?

No, un programa podría compilar y no hacer lo que se especifica

¿Si eliges una solución modularizada, esto te asegura que el programa es correcto?

No

¿Un programa correcto asegura eficiencia?

No

¿Un programa correcto asegura legibilidad?

No

¿Un programa bien documentado asegura corrección?

No

¿La adecuada utilización de variables locales y globales ayuda a la corrección de una solución?

No

¿Crees que existe una única solución correcta a un problema planteado?

No

¿Las **estructuras de datos elegidas** determinan que una solución sea **correcta** o no?

No

En todos los casos piensa los por qué.

Eficiencia
¿Cómo defines el concepto de eficiencia?

Se la define como la métrica de calidad de los algoritmos, asociada con una utilización óptima de los recursos del sistema de computo donde se ejecutará el programa, principalmente la memoria utilizada y el tiempo de ejecución empleado

¿Cómo puedes medir la eficiencia de una solución dada?

Calculando el espacio ocupado y el tiempo de ejecución

¿Existe una relación directa entre eficiencia y corrección? ¿Una solución eficiente es siempre correcta? Por qué.

Para que un algoritmo sea eficiente primero se debe haber comprobado que es correcto. Si, no podría ser eficiente sin primero ser correcta.

¿Podrías relacionar el concepto de Eficiencia con los siguientes ítems?

- Cantidad de líneas del programa
- Modularización
- Uso de variables locales
- Uso de determinadas estructuras de control
- Excesiva documentación de los algoritmos

- Cantidad de líneas del programa: No se puede establecer una relación directa
- Modularización: facilita el analisis de eficiencia, dado que puede calcularse el tiempo de ejecución de los distintos procesos uno a la vez, sin embargo una solución modularizada no es necesariamente más eficiente que una sin modularizar
- Las variables locales tienden a mejorar la legibilidad pero no hacen al programa más eficiente
- Las estructuras de control no afectan el tiempo de ejecución

- Un algoritmo bien documentado puede ser menos eficiente que otro

¿Cuáles crees que serían los ítems que influyen sobre el tiempo de ejecución de un programa y por qué?

a) Cantidad de datos de entrada
b) Cantidad de líneas de código
c) Cantidad de iteraciones presentes en el programa
d) Cantidad de variables declaradas

A. Si un programa se va a ejecutar solo con entradas “pequeñas” , la velocidad de crecimiento puede ser menos importante
B. No hay una relación directa
C. Las iteraciones multiplican el tiempo de ejecución del bloque
D. No lo afecta

Análisis comparativo de las estructuras de datos vistas

¿Qué diferencias conceptuales existe en las estructuras de datos vectores y listas?

Las dos estructuras son homogeneas
El acceso a los elementos en el vector es directo a través de un índice, en cambio en la lista es secuencial recorriendo los elementos de la lista
Las dos estructuras son lineales
Los vectores almacenan memoria continua en memoria y las listas se almacenan aleatoriamente, siguiendo un orden lógico
En vectores la ocupación en memoria se resuelve al momento de la **compilación** y en listas la ocupación en memoria se resuelve al momento de la **ejecución**
Los vectores poseen acceso directo, en cambio las listas son de acceso secuencial

Compara y explica detalladamente la operación de inserción y borrado en vectores y listas.

En los vectores al momento de insertar un elemento primero debe comprobarse que haya espacio, entonces se mueven todos los elementos desde la dimensión lógica hasta la posición donde se desea insertar el elemento, finalmente se asigna el elemento a la posición
En las listas no se controla el espacio en memoria, en cambio debe reservarse un lugar de alojado usando new() y generar el nuevo nodo, requiere menos accesos a memoria dado que al momento de la inserción simplemente se ajustan los punteros del nodo anterior (si no es el primero) y el puntero al siguiente elemento

Al borrar un elemento en un vector debe asignarse a cada elemento su siguiente desde la posición del elemento que deseo borrar hasta la dimensión lógica, finalmente reducir la dimensión lógica
Al borrar un elemento de una lista simplemente deben ajustarse los punteros para que el puntero anterior al nodo borrado (si no es el primero) apunte al nuevo nodo y el nuevo nodo apunte al de la posición actual

¿Cómo diferenciaría la búsqueda de un elemento en un vector y en una lista?

En los vectores hay que controlar que mientras se busca el elemento la dimensión lógica no exceda la dimensión física y avanzar aumentando una variable que sirva de índice
En las listas debe controlarse que no se llegue al último elemento de la lista y se recorre a través de los punteros al siguiente elemento, utilizando una variable auxiliar de tipo "lista"

¿Cómo es el acceso a un elemento conociendo su posición en un vector y en una lista?

En un vector el acceso es directo, vector[pos] mientras que en las listas el acceso es secuencial, es decir deben recorrer los componentes previos para llegar a la posición deseada

¿A igual cantidad de elementos en un vector y una lista qué diferencias hay respecto de la ocupación de memoria?

La diferencia es a favor de los vectores, ya que estos podrían almacenarse directamente en memoria en cambio en las listas se almacenará un puntero por cada componente además del elemento a guardar.