**1) Eficiencia**

**a-definir el concepto**

La eficiencia de un algoritmo estudia el tiempo de ejecución y la memoria que requiere para su ejecución.

**b-que se tiene en cuenta para analizar la eficiencia de un algoritmo?**

Para calcular la eficiencia, se calcula el tiempo que toma para ejecutarse el algoritmo, y además la cantidad de memoria que requiere para su funcionamiento.

**c-como se calculan uso de memoria y tiempo de ejecución (suele ser "calcular" y un programa)**

en cuanto a la memoria, esta se calcula en base a el costo de memoria que poseen sus variables utilizadas (en bytes). El tiempo de ejecución se calcula teniendo en cuenta que cada instrucción requiere cierta cantidad de tiempo por cada acción, siendo así, se utiliza la medida "udt" (unidad de tiempo).

**d-toda solución correcta es eficiente?**

No, porque la solución más allá de estar bien, puede no ser eficiente ya que el tiempo y la memoria que la solución utiliza puede ser mejorado en algunos casos.

**e-las estructuras de datos elegidas determinan que una solución sea eficiente?**

Si, ya que según el problema se determina que estructura utilizar para llegar a una solución más eficiente, como por ejemplo utilizar un vector en vez de una lista, cuando sabemos la cantidad de datos.

**f-un programa bien documentado asegura eficiencia?**

No, un programa bien documentado facilita la comprensión y mantenimiento del programa, pero esto no nos asegura eficiencia, esto depende del algoritmo y la implementación del código, no de su documentación.

**g-un programa modularizado asegura eficiencia?**

Si la modularizacion asegura eficiencia, ya que gracias a esto se pueden reutilizar los módulos y esto permite que se ahorre nuevas líneas de código y uso de memoria.

**2) Modularizacion y Parámetro**

**a-defina el concepto de modularizacion y sus ventajas**

La modularizacion es el proceso por el cual se agarran problemas del programa principal para utilizarlo en módulos aparte, para que se puedan llamar cuando los necesiten.  
La ventajas de modularizar son la reusabilidad de los módulos, ahorrando así líneas de código y uso de memoria.

**b-que módulos dispone pascal y en qué casos se usarían? (detalle al menos 3 diferencias)**

Pascal posee dos módulos, uno es la función (function) y otro es el procedimiento (procedure).  
-Una de las diferencias es que el procedimiento puede devolver más de una variable, en cambio la función solo devuelve una variable.  
-La función puede ser llamada dentro de un write, while, etc. Ahorrando así crear una variable para guardar ese valor  
-La función siempre tiene que devolver algo, el procedimiento, puede no devolver valores

**c-todo módulo procedure se puede escribir como una función?**

No, porque por ejemplo a veces el procedure devuelve dos variables cosa que la function no puede hacer.

**d-siempre se puede transformar un módulo de función en un módulo tipo procedure?**

Sí, todo módulo función se puede transformar en un procedure.

**e-analice los medios de comunicación entre los módulos**

Los medios de comunicación entre los módulos son los parámetros, que pueden ser por valor o por referencia, y las variables globales.

**f-defina alcance de una variable (¿que alcance tiene un parámetro?)**

una variable puede poseer un alcance local o global, aunque las únicas con carácter global son aquellas declaradas como variables globales, por otro lado, los parámetros poseen un carácter local, pudiendo solo utilizarse en el proceso en el que fueron declaradas o en aquellos procesos a los que entran como parámetros. En cambio, las variables globales pueden ser modificadas por todo el programa, sin necesidad de ser declaradas como parámetros en ningún momento.

**g-diferencias entre variable global y variable local**

Las variables globales pueden ser llamadas desde cualquier sin necesidad de ser declaradas. En cambio, las variables locales deben ser declaradas al comienzo de la ejecución de cada módulo y del Programa Principal.

**h-defina el concepto de parámetro en el contexto de la modularizacion**El parámetro es la forma en la que el programa principal y los módulos se comunican

**i-diferencias entre parámetros por valor y por referencia**

El parámetro por valor mantiene su valor en el módulo, más allá de que el modulo puede modificarlo a la hora de devolver el parámetro hacia el Programa principal, se devuelve su valor inicial. En cambio, el parámetro por referencia si se modifica su valor original y se devuelve modificado.

**j- ¿Por qué cree que es útil para el programador contar con los parámetros para escribir soluciones modularizado?**

Es útil para mantener la comunicación entre los módulos y el programa principal, facilitando así la modularizacion.

**k-que es la reusabilidad en la modularizacion?**

La reusabilidad, es la capacidad que poseen los módulos de ser utilizados nuevamente en otro momento ahorrando así nuevas líneas de código y uso de memoria.

**3) Estructura de Datos**

**a-Defina el concepto**.

Una estructura de datos es una forma de organizar y almacenar datos para que puedan ser utilizados eficientemente.

**b-realice un cuadro con los criterios de clasificación ubicando a las estructuras de datos vistas en el curso**

Durante el curso vimos 3 estructuras de datos , los registros, las listas y los vectores.  
Los registros poseen la capacidad de almacenar información de distintos valores (car, string, integer,etc) por lo que son comunmentes llamados por las otras estructuras de datos.  
Los vectores, a diferencia de los registros, solo pueden almacenar datos de 1 tipo (string, integer, char, registros, etc.), pese a esto, estos son capaces de mostrar la informacion almacenada en cualquier posición sin necesidad de recorrer los otros.  
Las listas poseen la capacidad de almacenar infinitamente datos, a diferencia de las otras dos estructuras de datos, el problema de estas es que en caso de requerir un dato, es necesario recorrer toda la lista hasta la posición en la que este ese dato.  
  
**b2-defina y diferencie las estructuras de datos: registro, vector y lista enlazada (dar más importancia a las 2 últimas)**

-Para acceder a un dato en el vector podemos accederlo directamente desde su posición, en cambio para acceder un dato en la lista necesitamos recorrerla hasta encontrarlo.

-Cuando declaramos un vector necesitamos saber la cantidad de elementos que puede llegar a tener el vector como máximo(dimL), en cambio cuando utilizamos la lista es porque no sabemos esa cantidad y son indeterminados los elementos que vamos a tener.  
-El vector almacena los datos en memoria contigua, en cambio la lista almacena los datos en nodos enlazados, no en memoria contigua.

**c-Plantee la operación de Inserción de un elemento en la estructura de datos lista y arreglo**

Arreglo  
1.Revisar si posee espacio para otro valor.  
2.En caso de ser posible mover el resto de valores hacia su casillero siguiente, para dejar "libre" el espacio donde este valor será introducido.  
3.Introducir el valor en el espacio.

Lista  
1.Despalzarse hasta la posición en la que se quiere insertar el dato.  
2.Modificar el valor siguiente como el dato a insertar.  
3.Modificar el valor siguiente del dato a insertar como el siguiente a la lista.

**d-realice un análisis del tiempo de ejecución de las operaciones implementadas en el punto c**

**e-diferencia entre dimensión lógica y física? justificar su uso**

La dimensión física es la cantidad de valores o datos que el arreglo puede almacenar esta se usa para saber hasta dónde cargar datos, mientras que la dimensión lógica es la cantidad de datos que el arreglo posee actualmente, esta se usa para que el recorrido del vector sea más eficiente.

**f-diferencia entre DISPOSE y NIL?**

Nil es un valor que se asigna al puntero para indicar que no apunta a ninguna dirección de memoria, se usa para inicializar punteros o indicar que un puntero esta vacío, Dispose es un procedimiento que se usa para liberar la memoria dinámica previamente asignada a un puntero con new.

**g-cuál es el riesgo de usar variables globales para la comunicación entre módulos?**

Al utilizar una variable global, se corre el riesgo de que algún proceso modifique esta variable por accidente, ocasionando la pérdida del valor que tenía con antes.

**g2-justifique porque es conveniente usar un parámetro por referencia en lugar de una variable global**

al utilizar un parámetro, se posee la capacidad de modificar la variable por valor sin miedo a que esta se modifique en el programa principal

**h-dadas dos variables A y B de tipo registro, indicar si las operaciones A:=B, A=B, READ(A), WRITE(B) son correctas (justificar)**

**A:=B** - La operación es válida, ya que ambos poseen el mismo tipo de dato.  
**A=B** - La operación No es válida, ya que la comparación entre registros debe ser dato por dato.  
**READ(A)** - La operación No es válida, la lectura de un registro debe ser campo a campo  
**WRITE(B)** - La operación No es válida, el imprimir un registro directamente no es posible, se imprime campo a campo.

**i-cuál es el objetivo de aplicar la algorítmica de corte de control sobre una estructura de datos**

El objetico de aplicar corte de control a una estructura de datos, es hacer que el algoritmo sea más eficiente, reduciendo el tiempo de ejecución al evitar procesar datos innecesarios.

**4) Otras**

**a-explique cuál es la importancia de que el lenguaje permita tipos de datos definidos por el usuario**

Hacer esto permite al programador el agilizar su programa, además de darle la posibilidad de utilizar variables con nombres cuya descripción sea clara, permitiéndole un mejor mantenimiento del código, así como un mejor entendimiento del mismo.

**b-qué diferencia hay entre un lenguaje "fuertemente" tipado con uno "dinámicamente" tipeado?**

Un lenguaje fuertemente tipado verifica y asegura que las operaciones solo se realicen entre tipos compatibles, proporcionando más seguridad y control en tiempo de compilación. En cambio, un lenguaje dinámicamente tipado no requiere especificar los tipos de las variables y realiza la comprobación de tipos en tiempo de ejecución, lo que ofrece mayor flexibilidad y facilidad de desarrollo, pero puede ser más propenso a errores en tiempo de ejecución.

**c-toda solución es correcta? Justifique**

Una solución es correcta si cumple con los requerimientos dados, para verificar estos requerimientos existen 4 métodos:  
- verificación, se trata de verificar si el programa cumple con estos requerimientos.  
- debuggin, se trata de la corrección de errores que el programa podría tener.  
- walkthrougs, son aquellas exposiciones del programa hacia un público determinado, buscando así errores en este.  
- testing, es el correr el programa, utilizando distintas variables posibles, así como las "variables limite" que el programa puede llegar a poseer.