1. Las instrucciones dentro de una estructura de control For siempre se ejecutan al menos una vez.
Rta: F, por ejemplo si el For es For i:= 10 to 1 do, ahi no se ejecutaria, por lo que no siempre se ejecutan.

2. Un programa modularizado es eficiente.
Rta: F, que sea modularizado no significa que sea eficiente ya que dentro de los módulos se pueden usar algoritmos menos eficientes, es decir, que el tiempo que tarda en ejecutarse y la memoria que requiera sea mayor que la del programa sin modularizar. Dato a tener en cuenta: Modularizar y eficiencia son conceptos distintos por lo que uno no afecta al otro si o si.

3. En el acceso a los campos de un registro es necesario respetar el orden en que fueron declarados.
Rta: F, el acceso a los campos de un registro es directo, por lo que se puede acceder sin necesidad de seguir un orden, solo se necesita especificar a que campo se quiere acceder.

4. Una variable global solo puede ser accedida y modificada desde el cuerpo del programa principal.
Rta: F, las variables globales también pueden ser accedidas desde módulos, de hecho es uno de los métodos de comunicación entre módulos.

5. Para usar una variable de tipo puntero p siempre se debe usar new(p).
Rta: se puede usar de todas maneras, por ejemplo se le puede asignar el valor nil u otra variable puntero.

6. Se pueden usar operaciones de entrada/salida sobre todos los campos de una variables de tipo registro.
Rta: F, solo se puede si el campo admire las operaciones de entrada salida, por ejemplo si un campo es de tipo puntero no se puede leer.

7. La estructura de datos lista es heterogenea.
Rta: F, todos los elementos son del mismo tipo.

8. Al asignar el valor de nil a un puntero se libera la memoria referenciada.
Rta: F, no se libera la memoria referenciada, se corta el enlace. Para liberarla es con dispose().

9. Un módulo función no puede retornar un tipo de dato puntero a un arreglo.
Rta: F, porque las funciones solo retornan tipos de datos simples y el puntero es uno de estos.

10. Si p es una variable de tipo puntero entonces no es valida la siguiente instruccion: readln(p^);
Rta: F, es válida siempre y cuando el tipo de dato lo permita, por ejemplo un integer.

11. Si un programa usa variables globales entonces no pueden contener modulos que usen parametros.
Rta: F, si pueden contener modulos que usen parametros, usar variables globales no impide su uso.

12. Un modulo con 5 lineas de codigo es mas eficiente en tiempos de ejecucion que un programa con 100 lineas de codigo.
Rta: F, la cantidad de lineas no esta relacionado con la eficiencia del programa, el programa de 5 lineas podria tener mas tiempo de ejecucion.

13. Un tipo de dato registro puede ser homogeneo si todos sus campos son del mismo tipo de dato.
Rta: F, es una estructura de datos heterogenea no importa que tenga los mismos tipos de datos en todos sus campos.

14. Nunca es posible acceder al nodo de la posicion N en una estructura de tipo lista.
Rta: F, si es posible en caso de guardar la direccion del nodo en una variable.

15. Un programa que contiene errores semánticos se puede compilar.
Rta: V, los errores semanticos no son evaluados por el compilador, se dan en tiempo de ejecucion.

16. Un programa solo permite variables globales cuyo tipo de dato es definido por el lenguaje.
Rta: F, tambien se pueden poner tipos de datos definidos por el programador.

17. Un vector siempre es almacenado en memoria estatica.
Rta: F, podemos declarar un puntero que apunte a un vector y de esa manera almacenar el vector en memoria dinamica.

18. Una variable de tipo puntero siempre se almacena en memoria dinamica.
Rta: F, la variable de tipo puntero se puede almacenar en la memoria estatica, lo que siempre se almacena en memoria dinamica es el tipo de dato apuntado por el puntero.

19. Una funcion puede retornar un tipo de datos lista, siendo lista un puntero a un nodo.
Rta: V, debido a que un puntero es un tipo de dato simple.

20. Antes de usar una variable puntero siempre se debe reservar en memoria.
Rta: F, se puede asignar otro puntero a la variable o asignarle nil.

21. La comunicacion mediante parametros asegura que un programa sea correcto.
Rta: F, que haya comunicacion mediante parametros no asegura que el programa ni pueda tener errores.

22. Siempre es posible eliminar el primer elemento es una lista.
Rta: F, ya que la lista puede estar vacia.

23. Las instrucciones dentro de una estructura de control repeat until se pueden ejecutar 0, 1 o mas veces.
Rta: F, la estructura de control repeat until se ejecuta siempre al menos una vez.

24. No es posible la utilizacion de las variables globales para la comunicacion entre modulos de un programa.
Rta: F, de hecho es uno de los metodos de comunicacion entre modulos junto con el pasaje de parametros.

25. Siempre es posible realizar la eliminacion de un elemento de un vector.
Rta: F, porque puede que el vector esté vacio y si es por posicion esta podria ser invalida.

26. Un programa modularizado puede no ser correcto.
Rta: V, que sea modularizado no significa que no pueda tener errores.

27. El acceso a un elemento de una estructura de datos lineal solo es posible a traves de un recorrido secuencial.
Rta: F, un vector es una estructura de datos lineal y no es necesario realizar un recorrido secuencial para acceder a un elemento.

28. En la tecnica de corrección de debugging es necesario analizar los casos limites del problema.
Rta: F, los casos limites se analizan en la tecnica de correccion Testing.

29. Un vector siempre se usa teniendo en cuenta la dimension logica.
Rta: F, si de antemano nos dicen que el vector va a ser usado en su totalidad no seria necesario tener en cuenta la dimension logica, ejemplo vector contador.

30. Una funcion puede devolver un tipo de dato registro, real, boolean, integer, etc.
Rta: F, solo devuelve tipos de datos simples, y un registro es un tipo de dato compuesto.

31. Un programa que solo usa variables globales no requiere modularizacion.
Rta: Es verdadero, porque el uso de variables globales no condiciona a que el programa requiera de modularizar, sino que la modularizacion es una practica empleada por el desarollador

32.Siempre es posible declarar un tipo subrango donde su tipo base sea cualquier de los tipos simples vistos en la teoria.
Rta: F, un real es un tipo de dato simple y no puede ser tipo base de un subrango, ya que los tipo base deben ser ordinales.

33.Incluir modulos dentro de un programa implica que el programa es mas eficiente que otro programa que realiza la misma tarea pero sin usar modulos.
Rta: F, la modularizacion y la eficiencia son conceptos diferentes, que la modularizacion exista no implica que un el programa sea mas eficiente que otro que no tenga modularizacion.

34.Si conozco la dimension logica de un arreglo de elementos enteros puedo utilizar un repeat until para recorrerlo e imprimir sus elementos.
Rta: F, en caso de que la dimension logica sea 0 y se usa un repeat until, como es una estructura de control pos condicional este se va a ejecutar minimo una vez y si la diml es 0, cuando se entre por primera vez al repeat se va a entrar a una pos del vector invalida.

35. La comunicacion entre el programa y modulos solo se puede hacer usando parametros.
Rta: F, tambien se puede hacer usando variables globales pero no es lo recomendable.

36.La sección TYPE de un programa únicamente se puede incluir luego de la declaración de las constantes y antes de la declaración de los módulos.
Rta: F. Los modulos tambien pueden tener type ya que son una unidad autocontenida.

37.Un programa que utiliza repeat until puede reescribirse usando un while.
Rta: V. como el repeat until se va a ejecutar minimo una vez, se puede reescribir con un while que tambien solo se ejecute una vez. En el caso contrario no se podria, ya que el while puede que no se ejecute nunca y el repeat si se ejecutaria una vez.

38.Para borrar un elemento en una pos válida del vector es necesario al menos realizar un desplazamiento de un elemento.
Rta: F. ya que el ultimo se puede eliminar solo disminuyendo la dimension logica o dimension fisica en uno.

39. todos los modulos de tipo funcion de un programa se pueden escribir como modulos de tipo procedimiento.
Rta: V. las funciones solo retornan un dato, y un proceso puede retornar 0,1 o mas datos. lo que no se puede es al reves.

40. se puede agregar un elemento al final de una lista sin necesidad de realizar un recorrido secuencial de la misma.
Rta: V. En caso de tener guardado en una var el puntero al ultimo nodo, se puede.

41. dado el encabezado siguiente: procedure prueba (a:integer;var b:real);
la sig invocacion es correcta: prueba(8,10);
Rta: F. Ya que el segundo parametro es pasado por referencia y para eso se debe pasar una variable cuando se invoca, no se puede un numero.

42. toda solución correcta es eficiente?
Rta: F. la eficiencia va de la mano con el estudio del uso del tiempo de ejecucion y uso de memoria de un programa, no si es correcto o no.

43.las estructuras de datos elegidas determinan que una solución sea eficiente?
Rta: F. que una solucion sea eficiente se determina por el tiempo de ejecucion y por la cantidad de memoria usada por el programa, no por las estructuras de datos que uses.

44. un programa correcto asegura la eficiencia?

Rta: F. que un programa sea correcto no quiere decir que sea eficiente. La eficiencia no estudia la correctitud, estudia el tiempo de ejecucion y uso de memoria de un programa.

45. un programa bien documentado asegura eficiencia?
Rta: F. la documentacion no va de la mano con eficiencia, la documentacion permite mayor legibilidad y comunicacion con otros programadores, la eficiencia mide el tiempo de ejecucion y uso de la memoria de un programa.

46. La tecnica de debugging se aplica en la etapa de diseño del algoritmo.
Rta: F. La tecnica de debugging se aplica en la etapa final, cuando el algoritmo se encuentra ya diseñado, de manera que se agregan sentencias adicionales para ver el comportamiento del programa.

47.Al pasar el puntero inicial de una lista por valor a un procedimiento, si dentro del mismo se modifica el contenido de los datos de la lista, una vez que el procedimiento termine, los datos de la lista no se veran modificados por fuera del procedimiento.
Rta: F. Si se veran modificados debido a que los nodos se encuentran en memoria dinamica por lo tanto si accedemos a ellos fuera del procedimiento los veremos modificados.

48.La estructura de control CASE, puede aplicarse a cualquiera de los tipos simples vistos en el curso, exceptuando los punteros.
Rta: F. Solo se aplica a los tipos de datos simples ordinales, y como el real es un tipo de dato simple hace que sea falsa la sentencia.

49.Pasar un parametro por valor siempre requiere mas memoria que pasar el mismo parametro por referencia.
Rta: F. En caso de pasar un char este va a ocupar 1 byte por valor y 4 bytes por referencia