Eficiencia

1. a) Se desea almacenar información sobre los últimos 12 resultados del campeonato mundial de Basketball. Si tuviera que ir guardando esta información en una estructura de manera tal de minimizar la cantidad de memoria utilizada: Seleccione la opción más adecuada y justifique.

Un vector
Una lista simple enlazada

Un vector sería la estructura más eficiente, dado que se conocen a priori la cantidad de elementos que almacenará y que utilizando una lista ocuparía espacio extra con los punteros

b) Compare la eficiencia de las operaciones de agregar, eliminar y buscar un dato en una lista simple cuando la misma está ordenada y cuando no lo está.

agregar es más eficiente cuando la lista está desordenada, ya que si estuviese ordenada necesitaría mantener ese orden ajustando los punteros por cada vez que debo agregar un elemento; en cambio si deseara agregar en una lista desordenada siempre podría agregar al principio o al final
eliminar tiene la misma eficiencia tanto en una lista ordenada como una desordenada, dado que en ambos casos tendrá el mismo trabajo de ajustar los punteros
buscar un dato es más eficiente en una lista ordenada, ya que permite realizar un corte de control para no tener que recorrer la lista por completo si el elemento no está en ella

En conclución una lista ordenada será más eficiente que una desordenada cuanto menor cantidad de veces que se actualize y mayor cantidad de veces se consulte

2. a) Mejore la eficiencia del siguiente bloque de código y justifique:

```
B:= 2;
C := 3;
For I:= 1 to 2000 do
   X := 2 * (B + C) * I;
```

```
B:= 2;
C := 3;
D := 2 * (B + C)
For I:= 1 to 2000 do
   X := D * I;
```

```
D := 2 * (2 + 3)
For I:= 1 to 2000 do
   X := D * I;
```

```
D := 2 * (2 + 3)
X := D * 2000;
```

```
X := 2 * (2 + 3) * 2000
```

En la primer mejora dejo el cálculo de 2 * (B + C) fuera del for, debido a que mejora el tiempo de ejecución sin afectar lo que el código hace.
En la segunda mejora elimino las variables B y C para ocupar menos lugar en memoria, esto solo puede hacerse si no son relevantes para el resto del programa
En la tercer mejora quito el for, debido a que no importa lo que suceda durante las iteraciones el resultado final siempre será el mismo y me está multiplicando el tiempo de ejecución del bloque
En la cuarta me deshago de la variable D para ahorrar memoria

3. Suponga que se pide crear una estructura de datos con 100 combinaciones diferentes de tres letras. Interesa que cada una de esas combinaciones se forme al azar. Ud. dispone de un procedimiento TRESLETRAS, que le devuelve un dato de tipo string con tres letras de la A a la Z mayúsculas al azar.
a) Resuelva lo pedido invocando al procedimiento TRESLETRAS, verificando que esa combinación ya no haya sido generada y guardándola en la estructura. Cuando ya esté generada, no la debe guardar.
b) A partir de la solución propuesta, realice un análisis de la misma desde el punto de vista de la memoria utilizada y del tiempo de ejecución empleado (calculando cantidad de operaciones efectuadas). Nota: el procedimiento TRESLETRAS no hay que implementarlo, solo se lo invoca.
c) Con respecto al inciso (a), para el proceso de búsqueda en la estructura elegida, ¿utiliza un algoritmo de recorrido secuencial?
d) ¿Como procede con la incorporación de nuevos datos a la estructura?

```
1   program ejTresLEtras;
2   {
13  type
14      vector = array[1..100] of String;
15
16  function noEstaGenerada(v : vector;s : string; dl : integer): boolean
17  var
18      aux : boolean; i:integer; //memoria: +1 unidad boolean + 1 unidad integer
19  begin
20      aux := false;   i := 1; // 2 unidades de tiempo
21      while (i < dl) and (aux = false)do //(entre 0 y 100) * 2 unidades de tiempo
22      begin
23          if (v[i] = s) then aux := True  // 2 unidades de tiempo //aplicar regla del if
24          else i := i+1;  //2 unidades de tiempo
25      end;
26  end;
27  var
28      s : string; v : vector;     //memoria : +1 unidad de string , +100 unidades de string
29      cont, dl : integer; //memoria: + 1 unidad integer + 1 unidad integer
30
31  begin
32      dl := 0; // 1 unidad T
33      cont := 0; // 1 unidad T
34      TRESLETRAS(s);
35      while cont < 100 do // N unidades de tiempo
36      begin
37          if noEstaGenerada(v,s,dl) then //N * Tiempo de noEstaGenerada * 1 unidad de tiempo
38          begin
39              dl := dl +1; //N * 2 unidades de tiempo
40              v[dl] := s; //N * 1 unidad de tiempo
41              cont := cont + 1; //N * 2 unidades de tiempo
42          end;
43      end;
44  end;
```

Memoria: 1 byte + 4 bytes + 256 (cantidad de caracteres +1) + 100 * 256 + 4 bytes + 4 bytes
Tiempo de ejecución: 2 T + N * T + N * Tiempo de función * T + 5N * T

Si utilizo un algoritmo de recorrido secuencial ya que no tengo otra opción, si la estructura estuviera ordenada por algún criterio podría realizar una búsqueda dicotómica para reducir el tiempo de ejecución

Dado que la estructura es un vector y que se que mi programa no puede almacenar más de 100 elementos no necesito controlar que la dimensión lógica supere la física, simplemente aumento la dimensión lógica y coloco el nuevo elemento en su posición