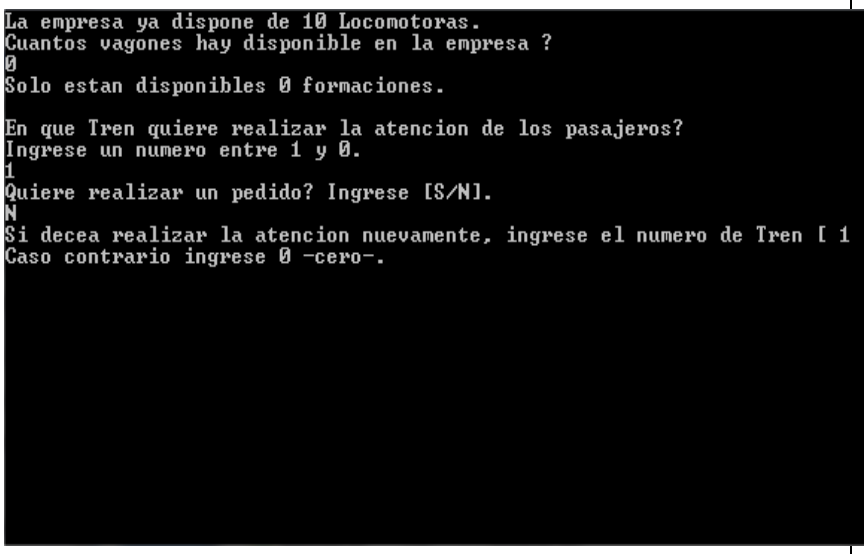
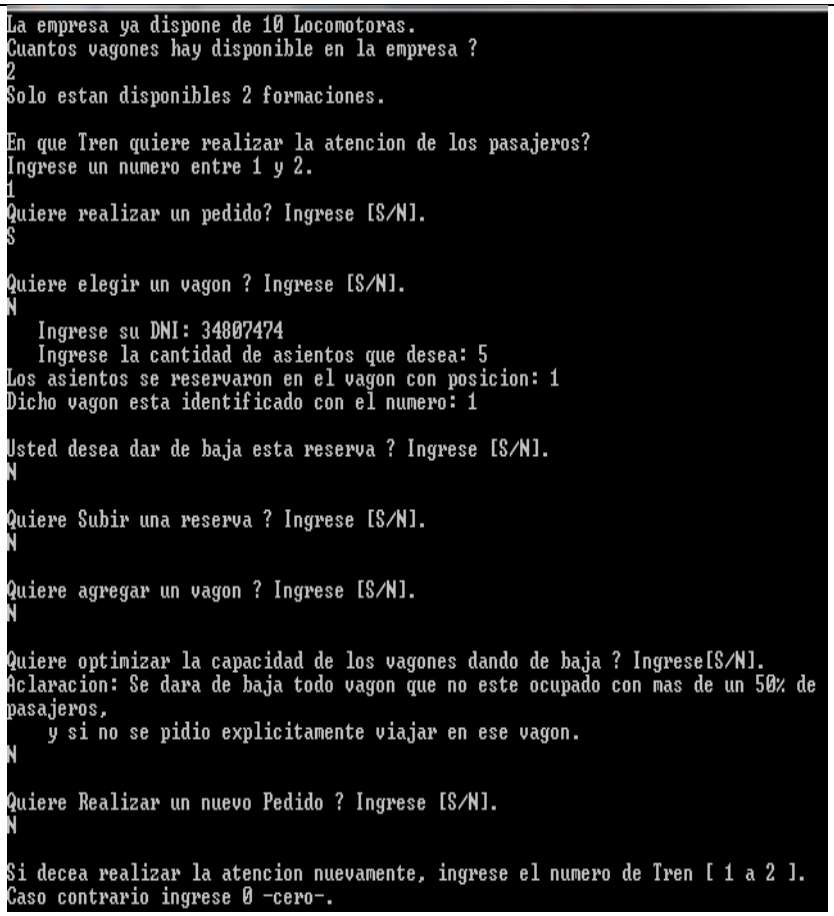
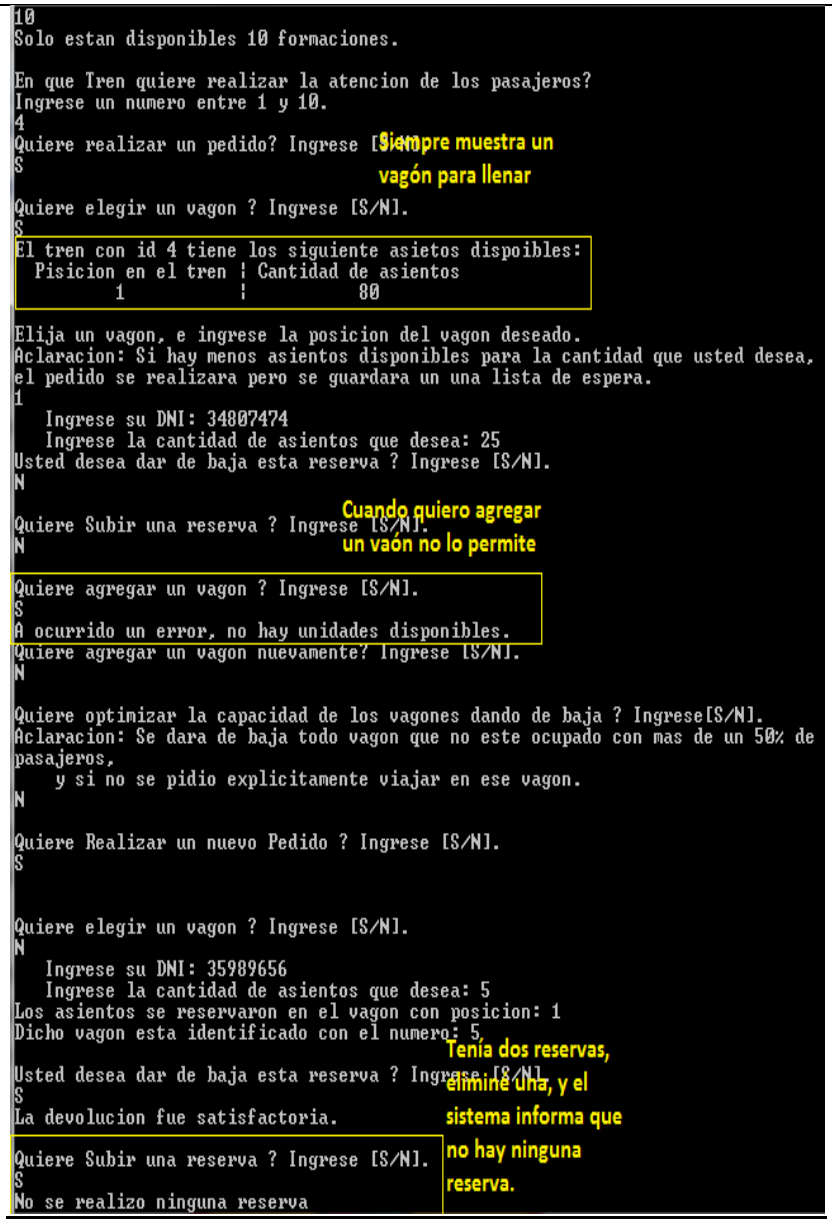
Fecha: 25/06/14

Corrección del EXE: BUENO Completar la columna de la derecha	
¿Funciona?	Si funciona, salvo algunas salvedades aclaradas más abajo.
Resultado de probar programa <u>Sin elementos.</u> **Deben colocar una imagen de la pantalla y justificar**	 Acá probé sin ninguna locomotora disponible, sin embargo me habilita una por defecto. Deberías restringir el uso del cero dejando como mínimo una locomotora disponible. O si no permitir que no haya locomotoras disponibles.

Resultado de probar programa <u>con 1 elemento</u> **Deben colocar una imagen de la pantalla y justificar**	 Elegí la disponibilidad de 2 locomotoras, hice una reserva sin ningún vagón en particular.

Resultado de probar programa con <u>Varios elementos</u> **Deben colocar una imagen de la pantalla y justificar**	 10 Solo estan disponibles 10 formaciones. En que Tren quiere realizar la atencion de los pasajeros? Ingrese un numero entre 1 y 10. 4 Quiere realizar un pedido? Ingrese [S/N]. S Quiere elegir un vagon ? Ingrese [S/N]. S El tren con id 4 tiene los siguiente asietos dispoibles: Pisicion en el tren ¦ Cantidad de asientos 1 ¦ 80 Elija un vagon, e ingrese la posicion del vagon deseado. Aclaracion: Si hay menos asientos disponibles para la cantidad que usted desea, el pedido se realizara pero se guardara un una lista de espera. 1 Ingrese su DNI: 34807474 Ingrese la cantidad de asientos que desea: 25 Usted desea dar de baja esta reserva ? Ingrese [S/N]. N Quiere Subir una reserva ? Ingrese [S/N]. N Quiere agregar un vagon ? Ingrese [S/N]. S A ocurrido un error, no hay unidades disponibles. Quiere agregar un vagon nuevamente? Ingrese [S/N]. N Quiere optimizar la capacidad de los vagones dando de baja ? Ingrese[S/N]. Aclaracion: Se dara de baja todo vagon que no este ocupado con mas de un 50% de pasajeros, y si no se pidio explicitamente viajar en ese vagon. N Quiere Realizar un nuevo Pedido ? Ingrese [S/N]. S Quiere elegir un vagon ? Ingrese [S/N]. N Ingrese su DNI: 35989656 Ingrese la cantidad de asientos que desea: 5 Los asientos se reservaron en el vagon con posicion: 1 Dicho vagon esta identificado con el numero: 5 Usted desea dar de baja esta reserva ? Ingrese [S/N]. S La devolucion fue satisfactoria. Quiere Subir una reserva ? Ingrese [S/N]. S No se realizo ninguna reserva Siempre muestra un vagón para llenar Cuando quiero agregar un vaón no lo permite Tenía dos reservas, elimine una, y el sistema informa que no hay ninguna reserva. Puse 10 locomotoras habilitadas, pero a medida que voy haciendo pedidos no puedo agregar más vagones en ninguna de ellas. Además tenía dos reservas, eliminé una y me dijo que no había ninguna más para subir.
Calidad general de los <u>mensajes por pantalla</u>	Muy buenos, faltarían corregir algunas expresiones sobre conceptos del programa que no coinciden con lo que hace la acción.
<u>Otros comentarios generales a la ejecución del programa</u>	El sistema por tener vagones únicos por locomotora como consecuencia la función de dar de baja algún vagón, no funciona.

Corrección del código

¿Usaron una lista para armar el tren?, en caso de respuesta negativa indicar que se usó	Si utilizaron una lista para armar el tren.
¿Usaron una matriz para cada vagón?, en caso de respuesta negativa indicar que se usó	Si utilizaron una matriz para armar cada uno de los vagones.
¿Cómo manejaron los valores N y P?	El P está manejado como un entero que es la dimensión física del arreglo pero utilizan el N como la dimensión lógica y no como la cantidad de vagones de los trenes.
¿Cómo resolvieron el tema de las filas A, B C y D de cada vagón?	Realizaron una matriz de 1..20 Filas y A..D columnas
¿Cómo se maneja la lista de espera?	Insertando al final de la lista y sacando adelante.
¿El TAD está conformado internamente con otro TAD?	NO, es un único TAD.
Revisar los cuidadosamente los valores que se pasan en los parámetros del TAD e indicar si se conserva el encapsulamiento de la estructura interna del TAD.	Si, se conserva el encapsulamiento, ya que sólo maneja tipos definidos por él en métodos internos del TAD.

Pegar código en Pascal, corregir en color ROJO y justificar en color VERDE

```
UNIT TipoTren;
INTERFACE
Type
Asiento = record {Cada asiento es un registro con las propiedades de si está libre o no}
        Libre:boolean;
        DNI:integer;
end;
Vagon = record
        Asientos: array [1..20,'A'..'D'] of Asiento; {Matriz de asientos}
{Acá tendrías que usar el tipo de dato Matriz que tenes declarado más abajo}
        Ocupantes:integer;
        Clientes_fijos:boolean; {True = hay clientes inamovibles, False = no los hay}
        idVagon:integer;
end;

Lista_Vagones = ^Nodo;
Nodo = record
        Datos_V:Vagon; {Información de cada vagón}
        Sig_Vagon:Lista_Vagones;
end;

Reserva = record
        DNI:integer;
        n_asientos:integer;
        idVagon:integer;  {inicializar en 0 'cero' si no se eligio ningun vagon}
end;
```

```
Lista_Reserva = ^Nodo_R;
        Nodo_R = record
        Datos_R:Reserva;
        Sig_R:Lista_Reserva;
end;

Tren = record
        idLocomotora:integer;
        L_V:Lista_Vagones;
        num_vagones:integer;
        L_R:Lista_Reserva;
        Recorrido:Lista_Vagones;
        Num_Recorrido:integer;
        Conjunto_Tren:set of 0..255; {los conjuntos tienen como max 255 enteros}
        {Acá tendrías que usar el Conjunto que tenes declarado abajo}
end;

Conjunto_Tren = set of 0..255; {*********}
Asientos= array [1..20,'A'..'D'] of Asiento; {*********}

Procedure Crear_Tren (var T:Tren; var Vagones_Totales:integer; idLocomotora:integer);
{Crea un Tren con vagones, en la cual no supera al maximo de vagones disponibles propuesta
por “Vagones_Totales”}

Procedure Agregar_Vagones (var T:Tren; var Vagones_disponibles:integer; var ok:boolean);
{Asigna un tren y lo engancha a el tren “T” si hay vagones disponibles en este caso “ok”
devuelve true, caso contrario false}

Procedure Dar_Baja_Vagon (var T:Tren; var Mensaje:string; var Vagones_Disponibles:integer);
{Da de baja un vagon si la capacidad no supera el 50%, si no hay solicitudes especificas de un
pasajero de querer viajar en ese vagon, y si hay lugares disponibles en ese vagon, devolviendo
el mensaje de baja del vagon.}

Procedure Subir_Reserva (var T:Tren; var DNI_R:integer; var idV:integer);
{Sube una reserva y devuleve  el DNI del pasajero de la reserva si se pudo realizar la misma y
el la idetificacion del vagon en la que fue reservado en “idV”, y si no devuelve DNI_R= -1}

Procedure Dar_Baja_Reserva (var T:Tren; Asientos:integer; idV:integer; DNI:integer; var
ok:boolean; var Mensaje:string);
{Se requiere para dar de baja cantidad de asientos en “Asientos”, identificacion de vagon en
“idV” para dar de baja los asientos que se habian reservado previamente, “ok” devuelve false
si ah ocurrido un error, en este caso en “mensaje” especifica el problema, caso contrario “ok”
devuelve true. }

Function Vagones_Pasajes_Completos (T:Tren) :integer;
{Dado un tren devuelve el número de vagones con pasaje completo }

Procedure Asignar_Tren (var T1:tren; T2:tren);
{Asigna el tren2 a tren1}
```

```
Procedure Realizar_Pedido (var T:Tren; Asi:integer; DNI:integer; var pos_Vagon:integer;
Fijo:boolean; var ok:boolean; var idV:integer);
{Con el DNI del cliente, la cantidad de asientos en “Asi”, la posicion del vagon “pos_Vagon”
(en caso de no elegir ningun vagon, pos_Vagon e idV devulve el vagon en la que fue
reservado),si pidio estar en ese vagon o no, si se pone en reserva el pedido “ok” devuelve
false caso contrario devuelve true, en “Fijo” si el pedido no se realiza por falta de asientos
dicho pedido se agregara a los pedidos de reserva.}

Procedure Inicializar_Recorrido (var T:Tren);
{Si se necesita saber datos de los vagones del Tren esta funcion inicializa el recorrido y asi
poder recorrerlo adecuadamente}

Procedure Informar_Pos_CantA (var T:Tren; var Pos:integer; var Cant_A:integer; var
ok:boolean);
{Despues de inicializar el recirrido esta funcion informa los Datos (Posicion del vagon,
Cantidad de asientos). Y cuando ya se informaron todos los vagones ‘ok” devuelve false, caso
contrario devuelve true (no es necesario inicializar la variable “ok”) }

Function ver_id_Locomotora (T:Tren):integer;
{informa la id de la Locomotora del tren “T”}

Implementation

{****  INICIO - Procedimientos Internos del TAD  ****}

Procedure Modificar_Asientos (var Asien:Asientos; DNI_R:integer; num_As:integer;
Accion:Char);
{PRE-Condicion: num de asientos disponibles del vagon >= a num de asientos a reservar}
{IMPORTANTE: “Accion” especifica si es Agregar <A> o Eliminar <E> reserva.  }
Var F:1..20; C:'A'..'D';
Begin
        F:=1;
        while (F <= 20) and (num_As > 0) do begin
                for C:='A' to 'D' do begin          {lenght no esta permitido en la práctica}
                        If (Accion = 'A') and (Asien[F,C].Libre = true) then begin
                                Asien[F,C].Libre:=false;
                                num_As:=num_As -1;
                                Asien[F,C].DNI:=DNI_R;
                        end else
                                If (Asien[F,C].DNI=DNI_R) and (Asien[F,C].Libre=false) and
                                (Accion='E') then begin
Asien[F,C].Libre:=true;
num_As:=num_As -1;
end;
                end;
                If (F < 20) then F:=F+1; {no importa que no llegue a 21 ya que num_As siempre
va a llegar a 0 cuando llegue al asiento 80}
        end;
End;

Procedure Eliminar_Vagon (var ant:Lista_Vagones; var act:Lista_Vagones);
Var aux:Lista_Vagones;
```

```
Begin
          aux:=act;
          ant^.Sig_Vagon:=act^.Sig_Vagon;
          act:=act^.Sig_Vagon;
          Dispose (aux);
End;

Procedure Eliminar_Reserva (var ant:Lista_Reserva; var act:Lista_Reserva);
Var aux:Lista_Reserva;
Begin
          aux:=act;
          ant^.Sig_R:=act^.Sig_R;
          act:=act^.Sig_R;
          Dispose (aux);
End;

Procedure carga_del_vagon (var reg: Vagon; cant_vagones:integer; var conj:Conjunto_Tren);
var
    i, id : integer;
    j : char;
Begin
   with reg do
   Begin
      For i :=1 to 20 do
       For j :='A' to 'D' do
         Begin
             Asientos[i, j].libre :=true;
             Asientos[i, j].DNI :=0; {Usamos 0 como dni nulo}
         end;
       ocupantes :=0;
       clientes_fijos :=false;
       id :=random(cant_vagones);
       While id in conj do
         id := random (cant_vagones);
       conj :=conj+[id];
       reg.IdVagon :=id;
   end;
End;
Procedure InsertarOrdAscendente (ptro_v:lista_vagones; var T:Tren);
VAR
    anterior,actual:lista_vagones;
Begin
    anterior:=T.L_V;
    actual:=T.L_V;
    While (actual<>nil) and (actual^.Datos_V.idVagon < ptro_v^.Datos_V.idVagon) do begin
       anterior:=actual;
       actual:=actual^.Sig_Vagon;
    end;
    If (anterior=actual) then begin
          ptro_v^.Sig_vagon:=anterior;
          T.L_V:=ptro_V;
    end else begin
```

```
          ptro_v^.Sig_vagon:=actual;
          anterior^.Sig_Vagon:=ptro_v;
    end;
End;

{****   FIN - Procedimientos Internos del TAD   ****}

Procedure Crear_Tren (var T:Tren; var vagones_totales:integer; idLocomotora:integer);
VAR
     ptro: Lista_Vagones;
     reg : Vagon;
Begin
  T.num_vagones :=0;
  T.L_R :=nil;
  T.Idlocomotora := idLocomotora;  {Le tenemos que pasar el numero del vector y listo}
  T.Conjunto_Tren:=[];  {inicializo el conjunto}
   If vagones_totales > 0 then
    Begin
       new(ptro);
       carga_del_vagon(reg,vagones_totales,T.Conjunto_Tren); {OJO acá estás mandando
vagones totales para usar en la asignación del random de los IDVagón cuando estás
decrementando dicha variable}
       {ACA FALTA INCREMENTAR T.num_vagones POR ESO SIEMPRE  TENES UNO SOLO}
       vagones_totales := vagones_totales -1;
       ptro^.datos_v:=reg;
       ptro^.Sig_vagon:=nil;
       T.L_V:=ptro;
    End;
End;

Procedure Agregar_Vagones (var T:Tren; var Vagones_disponibles:integer; var ok:boolean);
var
  ptro_v: lista_vagones;
  reg:vagon;
Begin
   if (Vagones_disponibles > 0) then {se tiene que poner el mismo nombre que el del
parametro} begin
      new (ptro_v);
      ok :=true;
      T.num_vagones := T.num_vagones +1;
      carga_del_vagon(reg,vagones_disponibles,T.Conjunto_Tren);
      ptro_v^.Datos_V :=reg;
      InsertarOrdAscendente(ptro_v,T);
    end Else
      ok :=false;
End;

Procedure Realizar_Pedido (var T:Tren; Asi:integer; DNI:integer; var pos_Vagon:integer;
Fijo:boolean; var ok:boolean; var idV:integer);
Var      aux:Lista_Vagones;
                  aux_R,nuevo:Lista_Reserva;
                  i, aux_pos:integer; {yo ya supongo que la posicion existe}
```

```
Begin
        aux:=T.L_V;
        aux_pos:=1;
        If (Fijo) then begin
                for i:=1 to (pos_Vagon - 1) do begin {se mueve hasta el vagon dado}
                        aux:=aux^.Sig_Vagon;
                        aux_pos:=aux_pos +1;
                end;
        end else begin
                while (aux <> nil) and ( (80 - aux^.Datos_V.Ocupantes) < Asi ) do begin
                        aux:=aux^.Sig_Vagon;
                        aux_pos:=aux_pos +1;
                end;
        end;
        If (aux <> nil) and ( (80 - aux^.Datos_V.Ocupantes) >= Asi ) then begin
                Modificar_Asientos (aux^.Datos_V.Asientos,DNI,Asi,'A');
                aux^.Datos_V.Ocupantes:=aux^.Datos_V.Ocupantes + Asi;
                pos_Vagon:=aux_pos;
                ok:=true;
                idV:=aux^.Datos_V.idVagon;
        end else begin
                ok:=false;
                new (nuevo);
                nuevo^.Sig_R:=nil;
                nuevo^.Datos_R.n_asientos:=Asi;
                nuevo^.Datos_R.DNI:=DNI;
                If (Fijo) then begin
                        nuevo^.Datos_R.idVagon:=aux^.Datos_V.idVagon;
                end else begin
                        nuevo^.Datos_R.idVagon:=0;
                end;
                If (T.L_R = nil) then begin
                        T.L_R:=nuevo;
                end else begin
                        aux_R:=T.L_R;
                        while (aux_R^.Sig_R <> nil ) do begin
                                aux_R:=aux_R^.Sig_R;
                        end;
                        aux_R^.Sig_R:=nuevo;
                end;
        end;
End;

Procedure Subir_Reserva (var T:Tren; var DNI_R:integer; var idV:integer);
Var     se_pudo:boolean; {cuando se realiza una reserva detiene el recorrido}
                auxV:Lista_Vagones;
                ant_R,act_R:Lista_Reserva;
Begin
        se_pudo:=false;
        auxV:=T.L_V;
        ant_R:=T.L_R;
        act_R:=T.L_R;
```

```
        DNI_R:=-1;
        while ( se_pudo = false ) and ( act_R <> nil ) do begin
                If (act_R^.Datos_R.idVagon <> 0) then begin {el cliente no pidio un vagon
especifico}
                        While (auxV <> nil ) and (act_R^.Datos_R.idVagon >
auxV^.Datos_V.idVagon) do begin
                                auxV:=auxV^.Sig_Vagon;
                        end;
                        if (auxV <> nil) and (act_R^.Datos_R.idVagon =
auxV^.Datos_V.idVagon) and ( (80 - auxV^.Datos_V.Ocupantes) > act_R^.Datos_R.n_asientos)
then begin
                                se_pudo:=true;
                                Modificar_Asientos
(auxV^.Datos_V.Asientos,act_R^.Datos_R.DNI,act_R^.Datos_R.n_asientos,'A');
                                auxV^.Datos_V.Ocupantes:=auxV^.Datos_V.Ocupantes +
act_R^.Datos_R.n_asientos;
                                auxV^.Datos_V.Clientes_fijos:=true;
                                DNI_R:=act_R^.Datos_R.DNI;
                                idV:=auxV^.Datos_V.idVagon;
                                Eliminar_Reserva (ant_R,act_R);
                        end;
      end else begin
                        while (se_pudo = false) and (auxV <> nil ) do begin
                                If ( (80 - auxV^.Datos_V.Ocupantes) >
act_R^.Datos_R.n_asientos )     then begin
                                        se_pudo:=true;
                                        Modificar_Asientos
(auxV^.Datos_V.Asientos,act_R^.Datos_R.DNI,act_R^.Datos_R.n_asientos,'A');

        auxV^.Datos_V.Ocupantes:=auxV^.Datos_V.Ocupantes + act_R^.Datos_R.n_asientos;
                DNI_R:=act_R^.Datos_R.DNI;
                                        idV:=auxV^.Datos_V.idVagon;
                                        Eliminar_Reserva (ant_R,act_R);
                                end;
                                auxV:=auxV^.Sig_Vagon;
                        end; {del while}
           end; {del If grande}
           auxV:=T.L_V;
                ant_R:=act_R;
                act_R:=act_R^.Sig_R;
        end; {del while grande}
End;

Procedure Dar_Baja_Reserva (var T:Tren; Asientos:integer; idV:integer; DNI:integer; var
ok:boolean; var Mensaje:string);
Var aux:Lista_Vagones;
Begin
        ok:=false;
        Mensaje:='No hay error.';
        aux:=T.L_V;
        While ( aux <> nil ) and ( idV > aux^.Datos_V.idVagon ) do begin
                aux:=aux^.Sig_Vagon;
```

```
        end;
        If (aux <> nil) and (idV = aux^.Datos_V.idVagon) then begin
                If (aux^.Datos_V.Ocupantes >= Asientos) then begin
                        Modificar_Asientos (aux^.Datos_V.Asientos,DNI,Asientos,'E');
                        aux^.Datos_V.Ocupantes:=aux^.Datos_V.Ocupantes - Asientos;
                        ok:=true;
                end else begin
                        Mensaje:='No hay suficientes asientos ocupados.';
                end;
        end else begin
                Mensaje:='El vagon no existe.';
        end;
End;

Function HayLugarEnTren (T:Tren; Ocupantes:integer; idVagon:integer):boolean;
Var LDisponibles:integer;
Begin
   LDisponibles:=0;
   while (T.L_V<>Nil) and (LDisponibles<Ocupantes) do begin
                if (T.L_V^.Datos_V.idVagon <> idVagon) then
                        LDisponibles:=LDisponibles + (80 - T.L_V^.Datos_V.Ocupantes);
                T.L_V:= T.L_V^.Sig_Vagon;
        end;
        If (LDisponibles >= Ocupantes) then begin
                HayLugarEnTren:=true;
        end else begin
                HayLugarEnTren:=false;
        end;
End;

Procedure Dar_Baja_Vagon (var T:Tren; var Mensaje:String; var
Vagones_Disponibles:integer);
var Aux,ant:Lista_Vagones; var F:1..20; var C:'A'..'D'; nue:Lista_Reserva;
cont_Reservas:integer; I:integer; X:integer; {el proceso Subir_Reserva demanda la variable 'X'
por eso la declaro}
Begin
    Aux:=T.L_V;
    ant:=T.L_V;
    Mensaje:='No se pudo hacer la baja de ningun vagon.';
    while (Aux<>Nil) do Begin
                cont_Reservas:=0;
                if (Aux^.Datos_V.Ocupantes<40) and (Aux^.Datos_V.Clientes_fijos=false) and
(HayLugarEnTren(T,Aux^.Datos_V.Ocupantes,Aux^.Datos_V.idVagon)) then begin
                        for F:=1 to 20 do begin
                                for C:= 'A' to 'D' do begin
                                        If (T.L_V^.Datos_V.Asientos[F,C].Libre=False) then
begin
                                                {Podrían tener una operación interna de
agregar reserva para no tener que crear la lista sobre el mismo procedimiento en que haces la
lógica de otra funcionalidad}
                                                new (nue);
                                                nue^.Datos_R.DNI:=
```

```
T.L_V^.Datos_V.Asientos[F,C].DNI;
                                                nue^.Datos_R.n_asientos:=1;
                                                nue^.Datos_R.idVagon:=0;
                                                nue^.Sig_R:=T.L_R;
                                                T.L_R:=nue;
                                                cont_Reservas:=cont_Reservas+1;
                                        end;
                                end;
                        end;
                        Mensaje:='Se da de baja el Vagon.';
                        Eliminar_Vagon (ant,Aux);
                        Vagones_Disponibles:=Vagones_Disponibles +1;
                        for I:=1 to cont_Reservas do begin
                                Subir_Reserva (T,X,X);
                        end;
                end;
                ant:=Aux;
                Aux:=Aux^.Sig_Vagon;
        end;
End;

Function Vagones_Pasajes_Completos (T:Tren):integer;
Var Total:integer;
Begin
        Total:=0;
        while (T.L_V <> nil) do begin
                If ( T.L_V^.Datos_V.Ocupantes = 80 ) then begin
                        Total:= Total + 1;
                end;
                T.L_V:=T.L_V^.Sig_Vagon;
        end;
        Vagones_Pasajes_Completos:=Total;
End;


Procedure Asignar_Tren (var T1:tren; T2:tren);
Begin
        T1:=T2;
End;

Procedure Inicializar_Recorrido (var T:Tren);
Begin
        T.Recorrido:=T.L_V;
        T.Num_Recorrido:=0;
End;

Procedure Informar_Pos_CantA (var T:Tren; var Pos:integer; var Cant_A:integer; var
ok:boolean);
Begin
        If ( T.Recorrido = nil ) then begin
                ok:=false;
        end else begin
```

```
                ok:=true;
                T.Num_Recorrido:=T.Num_Recorrido +1;
                Pos:=T.Num_Recorrido;
                Cant_A:=(80 -T.Recorrido^.Datos_V.Ocupantes);
                T.Recorrido:=T.Recorrido^.Sig_Vagon;
        end;
End;

Function ver_id_Locomotora (T:Tren):integer;
Begin
        ver_id_Locomotora:=T.idLocomotora;
End;
END.
Program project1;

Uses
        TipoTren;
Const
        P = 10;

Type
Locomotoras = array [1..P] of Tren; {Array de locomotoras}

Var     L:Locomotoras;
                N_Vagones, DimL,i:integer;

Procedure Informar_Datos (T:Tren);
{informa los datos para que despues el cliente elija el vagon}
var Pos,Cant_A:integer; ok:boolean;
Begin
        Inicializar_Recorrido (T);
        Informar_Pos_CantA (T,Pos,Cant_A,ok);
        Writeln ('El tren con id ',ver_id_Locomotora(T),' tiene los siguiente asietos dispoibles:
');
        Write('  '); Write('Pisicion en el tren'); Write(' | '); Writeln('Cantidad de asientos');
        while (ok) do begin
                write('         '); Write(Pos); Write('          |          '); Writeln(Cant_A);
                Informar_Pos_CantA (T,Pos,Cant_A,ok);
        end;
End;

Procedure Crear_Sistema_de_Trenes (var L:Locomotoras; var DimL:integer; var N:integer);
Begin
        While ( DimL < P ) and ( N > 0 ) do begin
                DimL:=DimL +1;
                Crear_Tren (L[DimL],N,DimL);
        end;
        Writeln ('Solo estan disponibles ',DimL,' formaciones.');
        Writeln;
End;
```

```
Procedure Leer_SN (var SN:char);
Begin
        Readln (SN);
        While (SN <> 'S') and (SN <> 'N') do begin
                Writeln ('Ingrese la tecla -S- o -N- .');
                Readln (SN);
        end;
End;

Procedure Atencion (var T:Tren; var N:integer);
var     SN:char;{SN lo uso para determinar lo que quiere el usuario }
        Fijo,ok:boolean;
        Asient,DNI,pos_Vagon,idV:integer;
        Mensaje:string;
Begin
        Writeln ('Quiere realizar un pedido? Ingrese [S/N].');
        Leer_SN (SN);
        While (SN = 'S') do begin
         Writeln;
                Writeln ('Quiere elegir un vagon ? Ingrese [S/N].');
                Leer_SN (SN);
                If ( SN = 'S') then begin
                        Informar_Datos (T);
                        Writeln;
                        Writeln('Elija un vagon, e ingrese la posicion del vagon deseado.');
                        Write ('Aclaracion: Si hay menos asientos disponibles para la
cantidad');
                        Write (' que usted desea, el pedido se realizara pero se guardara un
una');
                        Writeln (' lista de espera.');
                        Readln (pos_Vagon);
                        Fijo:=true; {ya que se eligio un vagon}
                end else begin
                        Fijo:=false;
                end;
                Write ('   Ingrese su DNI: '); Readln (DNI);
                Write ('   Ingrese la cantidad de asientos que desea: ');Readln(Asient);
                Realizar_Pedido (T,Asient,DNI,pos_Vagon,Fijo,ok,idV);
                If (ok) then begin { si el pedido se agregaro correctamente a el vagon}
                        If (Fijo=false) then begin
                                Writeln('Los asientos se reservaron en el vagon con posicion:
',pos_Vagon);
                                Writeln ('Dicho vagon esta identificado con el numero: ',idV);
                                Writeln;
                        end;
                        Writeln ('Usted desea dar de baja esta reserva ? Ingrese [S/N].');
                        Leer_SN (SN);
                        while (SN ='S') do begin
                                Dar_Baja_Reserva (T,Asient,idV,DNI,ok,Mensaje);
                                If (ok = false) then begin
                                        Writeln (Mensaje,' , lo quiere realizar devuelta ?');
                                        Leer_SN (SN);
```

```
                                    end else begin
                                            SN:='N';
                                            Writeln ('La devolucion fue satisfactoria.');
                                    end;
                            end;
                    end else begin
                            Writeln ('Su pedido se guardo en una lista de espera.');
                    end;
            writeln;
                    Writeln ('Quiere Subir una reserva ? Ingrese [S/N].');
                    Leer_SN (SN);
                    If (SN = 'S') then begin
                            Subir_Reserva (T,DNI,idV);
                            If (DNI <> -1) then begin
                                    Writeln ('Se realizo la reserva del cliente con DNI: ',DNI,'.');
                                    Writeln ('En el vagon numero ',idV,'.');
                            end else begin
                                    Writeln ('No se realizo ninguna reserva');
                            end;
                    end;
            Writeln;
                    Writeln ('Quiere agregar un vagon ? Ingrese [S/N].');
                    Leer_SN (SN);
                    while (SN = 'S') do begin
                            Agregar_Vagones (T,N,ok);
                            if (ok) then begin
                                    Writeln ('Se agrego correctamente.');
                            end else begin
                                    Writeln ('A ocurrido un error, no hay unidades disponibles.');
                            end;
                            Writeln ('Quiere agregar un vagon nuevamente? Ingrese [S/N].');
                            Leer_SN (SN);
                    end;
            Writeln;
                    Writeln ('Quiere optimizar la capacidad de los vagones dando de baja ?
Ingrese[S/N].');
                    Writeln ('Aclaracion: Se dara de baja todo vagon que no este ocupado con
mas de un 50% de pasajeros,');
                    Writeln ('    y si no se pidio explicitamente viajar en ese vagon.');
                    Leer_SN (SN);
                    If (SN = 'S') then begin
                            Dar_Baja_Vagon (T,Mensaje,N);
                            Writeln (Mensaje);
                    end;
            Writeln;
                    Writeln ('Quiere Realizar un nuevo Pedido ? Ingrese [S/N].');
                    Leer_SN (SN); {Para el loop del while}
            Writeln;
           end;
End;
```

```
BEGIN {Del programa principal}
        DimL:=0;
        Writeln ('La empresa ya dispone de ',P,' Locomotoras.');
        Writeln ('Cuantos vagones hay disponible en la empresa ?');
        Readln (N_Vagones);
        Crear_Sistema_de_Trenes (L,DimL,N_Vagones);
        Writeln ('En que Tren quiere realizar la atencion de los pasajeros? ');
        Writeln ('Ingrese un numero entre 1 y ',DimL,'.');
        Readln (i);
        {Qué pasa si yo ingreso cero la primera vez? Porque no usar un while?}
        Repeat
                Atencion (L[i],N_Vagones);
                Writeln ('Si decea realizar la atencion nuevamente, ingrese el numero de Tren
[ 1 a ',DimL,' ].');
                Writeln ('Caso contrario ingrese 0 -cero-.');
                Readln (i);
        Until ( i = 0 );
End.
```

Otros comentarios al código: Faltan realizar/ separar algunas cuestiones de funcionalidades dentro de un mismo procedimiento, porque se hace difícil la lectura y el seguimiento del código si se resuelven varias cosas juntas en una misma operación, sobre todo en los procesos del TAD.