APELLIDOS: ***NOMBRE:***
TITULACIÓN: ***__GII __GII+ADE __GII+CRI __GII+MAT __GIS __GIS+GII __GIS+MAT***
DNI:
Duración: 1:45 h.
Calificación:

Se recuerda que, según la guía docente, el 25% se corresponde con prueba escrita teórica y 60% con prueba escrita práctica. El 15% restante proviene de las pruebas prácticas realizadas a lo largo del curso. Se sugiere dejar 1 hora de esfuerzo en la resolución del ejercicio 5. Contestar en la hoja de enunciados TODOS los ejercicios.

1) (**Prueba escrita teoría, 0.5 punto)** Describir el proceso de inserción, en un árbol AVL, de la siguiente secuencia de nodos: 10, 12, 14, 9, 11, 13, 19, 16, 17. Se deberán mostrar **todos los pasos** intermedios, indicando y describiendo las acciones que se deban ir tomando.

2) (**Prueba escrita teoría, 1 punto**).
   a. (**0.5 puntos**) Especifique algebraicamente la operación Frontera de árboles binarios teniendo en cuenta las operaciones constructoras generadoras:
      CrearArbolBinVacio: → TArbolBinario
      ConstruirArbolBin: TArbolBinario X TElemento X TArbolBinario → TArbolBinario
      Así como las operaciones del TAD Lista:
      CrearListaVacia: →TLista
      Construir: TElemento X TLista → TLista
      Concatenar: TLista X TLista → TLista

   b. (**0.5 puntos**) Dado un grafo dirigido en memoria dinámica y dos vértices, si se quiere implementar una operación para devolver la longitud de un camino posible entre ambos vértices, cuántos tipos abstractos de datos necesitaría en unidades diferentes. Justifique las respuestas e indique brevemente cómo se haría.

3) (**Prueba escrita teoría, 1 punto**) Calcule, mediante el método de expansión de recurrencias visto en clase, la complejidad algorítmica en notación O() del algoritmo de Búsqueda Binaria, mediante el cual se busca un elemento en una lista de valores ordenados.

4) (**Prueba escrita práctica, 1,5 puntos**) Sobre árboles binarios:

   a. **[0,75 puntos]** Implemente en una unidad la definición de tipos de un árbol binario en memoria dinámica que guarde elementos de tipo TElemento definido en la unidad uElemento. A continuación implemente la unidad uElemento con la definición del tipo TElemento como un array con 2 enteros. Implemente los subprogramas adecuados en sus respectivas unidades y las llamadas necesarias en el programa principal para mostrar por pantalla todos los elementos que contenga un árbol (cada elemento en una línea) que previamente se ha inicializado (no se pide inicializar el árbol).

b. **[0,75 puntos]** Haciendo uso del TAD ÁrbolBinario y el TAD Lista vistos en clase, implemente el subprograma `PadresNoAbuelos` que dado un árbol binario devuelva una lista con los elementos que pertenecen a los nodos del árbol que sean padres, pero no abuelos. Asumimos que se dispone de la implementación de la función `EsHoja`, que recibe un árbol y devuelve un booleano. Se entiende que el tipo base del arbol binario y de la lista es el mismo `TElemento`.

5) (Prueba escrita práctica, 4,5 puntos)

Una universidad quiere limpiar su imagen pública recientemente afectada por un caso de fraude. Su equipo de gobierno persigue que los egresados de la universidad tengan un nivel medio superior al ya de por sí excelente. Para ello, ha creído conveniente crear un sistema (a coste casi cero) de lecciones online de refuerzo obligatorias para los alumnos más rezagados, y optativas para el resto. Cada curso académico, un sistema informatizado calculará el Nivel de Esfuerzo Personal (NEP) de cada uno de sus alumnos, y señalará anónimamente quiénes de ellos tendrán que asistir obligatoriamente a las clases de refuerzo. Para ello, se aportará información, tanto por el propio estudiante como por los profesores, que se utilizará para medir el rendimiento de cada alumno más allá de las calificaciones que obtiene, ya que, como todo el mundo sabe, las calificaciones pueden venir sesgadas por la facilidad de algunas asignaturas, del plan de estudios, de los docentes contratados u otros factores de dudosa objetividad. Para la versión 1.0 del cálculo del NEP, el vicerrector competente ha diseñado una combinación de entradas que incluye información sobre las 4 asignaturas más importantes de cada curso (2 por cuatrimestre) para cada titulación (elegidas por los docentes y vulgarmente llamadas asignaturas *estrella*) así como la información del entorno del estudiante, formado por los 5 compañeros que cada estudiante decide elegir como Entorno Personal, y un factor que evalúa el progreso del estudiante durante el curso si durante el primer cuatrimestre le fue mal.

La información que entra como entrada en el cálculo del NEP es: Suma de Desviaciones de las calificaciones del estudiante en las asignaturas estrella respecto a las Notas Medias de cada asignatura (SDNM); lo mismo pero teniendo solamente en cuenta su Entorno Personal (SDNM_EP); y la EVOlución del estudiante durante el curso (EVO).

El SDNM se calcula como la suma de las diferencias entre las calificaciones obtenidas por el alumno con respecto a cada una de las medias de las asignaturas estrella en su curso académico. Por ejemplo, para un estudiante que termina 1º de GII, y si las asignaturas estrella son MDA, IP, ED y CAL con notas medias para su clase de 5.6, 4.7, 4.1 y 5.4, y calificaciones personales de 3.8, 4.2, 3.5 y 7.6, su SDNM se calcularía como (3.8-5.6)+(4.2-4.7)+(3.5-4.1)+(7.6-5.4) = (-1.8) +(-0.5) + (-0.6) + (2.2) = -0.7

Para el cómputo del SDNM_EP se procedería del mismo modo con las 4 asignaturas estrella de cada uno de los 5 estudiantes del entorno personal y se calcularía la media de los 5 resultados.

Para el cómputo de la EVO se calcularía la SDNM de las asignaturas estrella del primer cuatrimestre (SDNM_1) y si devuelve positivo, su EVO es directamente 0, mientras que si devuelve un valor negativo (dando a entender que le fue peor que a la media de la clase), se calcula la SDNM de las asignaturas estrella del segundo cuatrimestre (SDNM_2) y se compara con la mitad de la SDNM_1, si es igual o mayor, el EVO sería +1, y en caso contrario -1.

$$\text{EVO} = \begin{cases} -1 \; si \; (SDNM\_1 < 0) \; Y \; (SDNM\_2 < 0.5 * SDNM\_1) \\ 0 \; si \; (SDNM\_1 > 0) \\ +1 \; si \; (SDNM\_1 < 0) \; Y \; (SDNM\_2 \geq 0.5 * SDNM\_1) \end{cases}$$

Con los valores del ejemplo anterior, tendríamos que el estudiante de referencia habría obtenido un SDNM_1= (-1.8)+(-0.5) = -2.3, negativo, por lo que habría que calcular su SDNM_2 = (-0.6) + 2.2 = 1.6, siendo SDNM_2 > 0.5 * SDNM_1 (1.6 > -1.15) por lo que su EVO sería +1.

Finalmente, la expresión del NEP es: NEP = SDNM + SDNM_EP + EVO

Se pide:

a) **[0,5 puntos]** Diseñe la estructura de datos `TNEP_Universidad` para cargar toda la información del NEP de sus alumnos en memoria principal. La estructura es un listado de Centros de tipo `TCentro` (los centros son las Escuelas/Facultades). Cada Centro, además del nombre del mismo (`string`), contiene información de las titulaciones (`TTitulacion`) que pertenecen al centro, y en cada titulación se guarda un entero con el código de la titulación y el listado de los estudiantes de la titulación. La información concreta de cada estudiante (`TPerfilEstudiante`) es: código de estudiante (compuesto por 2 enteros, `TCodigoEstudiante`), un entero para describir el último curso de asignaturas estrella completado por el estudiante (`ultimoCurso`), otro entero para saber en cuántos grupos se ha elegido al alumno como parte de un Entorno Personal (`numSeleccionEP`), un campo para guardar su NEP actual (valor de tipo real), un almacen con los 5 códigos de estudiante de su Entorno Personal, y un almacen con la información de las 4 asignaturas estrella (las 2 primeras correspondientes a las de primer cuatrimestre y las 2 segundas a las de segundo cuatrimestre). La información de cada asignatura estrella que guarda un alumno será el código de asignatura (un valor entero) y la calificación final (valor real) obtenida en dicha asignatura.

b) **[1 punto]** Crear los procedimientos adecuados para insertar en la estructura principal un alumno de nuevo ingreso al que en el proceso de matrícula le han asignado código de estudiante y código de titulación que se pasarán a los subprogramas adecuados. Algunos de los campos de la información de su perfil quedarán vacíos hasta que no complete el primer curso, haya seleccionado a su Entorno Personal y haya cursado las asignaturas estrella.

c) **[1 punto]** Para asegurar que no se eligen siempre a los mismos alumnos en los Entornos Personales, no está permitido que un alumno pertenezca a más de 10 grupos. Crear el subprograma `Verificacion` para comprobar que, dada una variable con toda la información (de tipo `TNEP_Universidad`), no hay estudiantes elegidos en más de 10 grupos. Para ello se tendrá en cuenta que cada vez que es elegido un estudiante en un Entorno Personal, se incrementa su correspondiente campo `numSeleccionEP` de número de Entornos Personales en los que aparece. Cada vez que se detecte que un alumno aparece en más de 10 grupos se enviará una advertencia al gestor del vicerrectorado de alumnos mediante la llamada al procedimiento de prototipo:

```
PROCEDURE AvisoViceAlumnos(cod: TCodigoEstudiante; num: integer);
{PRE: "cod" es el código de estudiante, y "num" es el número de
apariciones en Entornos Personales que tiene}
```

d) **[2 puntos]** Dada la estructura principal de tipo `TNEP_Universidad` y un código de estudiante (`TCodigoEstudiante`), obtener el NEP que le corresponde para el curso académico 2017/18 y asignárselo a su campo específico. La información de las asignaturas de los respectivos planes de estudio se mantiene en un único fichero que custodia el Vicerrectorado de Ordenación Docente. El vicerrectorado ofrece una función para extraer la nota media de una asignatura concreta calculada para todo el alumnado que la cursó en un curso académico y tiene el siguiente prototipo:

```
FUNCTION NotaMediaAsign(codigoAsignatura, curso: integer):real;
{PRE: codigoAsignatura es un entero que identifica la asignatura, y
curso es un entero; por ejemplo 1516 para identificar el curso
2015/16.
POST: la función devuelve la nota media de la asignatura para el
curso dado}
```

Además, para la localización de alumnos en el listado de estudiantes de la titulación se ofrece el uso del procedimiento `LocalizaEstudiante` con prototipo:

```
PROCEDURE LocalizaEstudiante(cod: TCodigoEstudiante; lEst:
TListaEstudiantes; VAR pos: TListaEstudiantes);
{PRE: "cod" es el código de estudiante y "lEst" la lista de
estudiantes de una titulación
POST: "pos" es un puntero que apunta al nodo que guarda el perfil
del alumno con código de estudiante "cod". Devuelve NIL en caso de
no encontrar ese código de estudiante en la lista}
```