UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 23/02/2017

Apellido y nombre:_____ Legajo: _____ Cursó con Prof: _____

- Si luego de la lectura del examen, durante la resolución tiene alguna duda, escriba hipótesis de trabajo, las cuales también serán evaluadas.
- Puede desarrollar la algoritmia en C, ó C++ o mediante diagramas.
- Condición de aprobación: resolución correcta de la función requerida y la respuesta correcta, al menos, a una afirmación y su justificación.

Regularización – Promoción Programación[1]

Temas evaluados: Comprensión de situación problemática, funciones, estructuras de datos enlazadas y contiguas, flujos y archivos.

Contexto

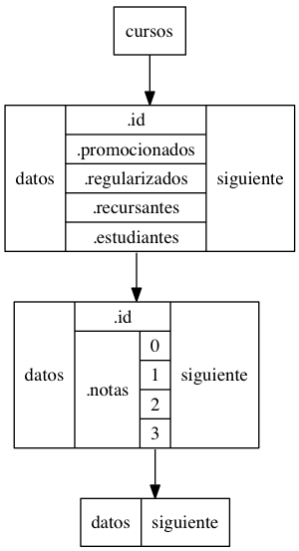
Usted es parte del equipo de desarrollo de la Secretaría Académica de una Universidad Nacional. El equipo debe determinar, en función de las evaluaciones del presente año lectivo, la situación de regularización o eventual aprobación directa (*promoción)* según la normativa propia de cada cátedra para ese fin. En esta situación particular se deben analizar los datos de los estudiantes cursantes en una materia de programación.

Requerimientos para regularización o promoción de Algoritmos y Estructura de Datos.

1. Durante el curso hay cuatro evaluaciones: dos parciales, un TP, y un coloquio oral. Cada evaluación tiene tres oportunidades, la original y dos recuperatorios.
2. La nota que vale es la de la última oportunidad rendida, más allá que no sea la más alta, ni siquiera la aprobada.
3. Un estudiante promociona con las cuatro evaluaciones con nota igual o mayor a 8, regulariza con nota mayor o igual a 6, y recursa con nota menor a 6.

Definición del problema

Se necesita procesar las novedades de un archivo que contiene las notas de estudiantes de diferentes cursos, para luego calcular estadísticas de cada curso.



Se pide

1. Codifique la declaración de la variable **cursos** y declare los tipos de datos necesarios para declararla:
   a. La variable **cursos** es un puntero al primer nodo de una **lista enlazada de cursos**. Cada curso tiene el **id** del curso, la cantidad de **promocionados**, **regularizados**, y **recursantes**.
   b. Cada curso, también tiene la lista de **estudiantes**, donde, a su vez, cada estudiante tiene un **id** de estudiante y un arreglo de **evaluaciones**, de longitud cuatro, uno para cada evaluación.
   c. Por último, cada evaluación es una lista enlazada con las **notas** de la evaluación y sus recuperatorios; es decir, la lista tiene longitud entre cero y tres.

   Se adjunta esquema orientativo.
2. Codifique o diagrame una función **ProcesarNovedades(nombreDeArchivo, cursos)** que lee de un archivo registros con el id del **curso**, el id del **estudiante**, el **parcial** (1, 2, 3, ó 4), y la **nota**, y utiliza la función **AgregarNota** para registras las novedades en la lista cursos. Una nota cero equivale a un ausente. Los registros están parcialmente ordenados, para un estudiante y para una evaluación determinada, la primera nota es la de la instancia original, la segunda es la del recuperatorio, y la tercera la del segundo recuperatorio. No hay otro orden definido.
3. Codifique o diagrame la función **AgregarNota(cursos, curso, estudiante, parcial, nota)** que dada la lista de **cursos,** busca en un **curso** a un **estudiante**, y para un **parcial** le agrega la **nota**. Si no había nota la agrega, si ya había una nota, la nueva es en concepto de primer recuperatorio, si había dos, es en concepto de segundo recuperatorio. Precondiciones: el curso, el estudiante, y el parcial existen, y no se van a agregar más de tres notas por parcial.
4. Codifique o diagrame la función **CalcularEstadísiticas(cursos)** que registra en cada curso la cantidad de **promocionados**, **regularizados**, y **recursantes**. La nota que vale es la de la última oportunidad de cada evaluación. Para determinar el estado invoque a **GetEstado** que dado el arreglo de evaluaciones de un estudiante retorna 1 si promociona, 2 si regulariza, 3 si recursa.
5. Punto extra opcional: Codifique la función **GetEstado**.

[1] Se aclara que todo lo expuesto es al solo efectos de una simulación para el examen que se evalúa, no establece directiva real alguna de ninguna materia en particular.

Puntaje

1. 2 puntos.
2. 2 punto.
3. 2 puntos.
4. 3 puntos.
5. 1 punto.

Para aprobar, por lo menos el 50% de los ítems 1 a 4 debe estar bien resuelto.

Resolución

```
1.
struct Nota {
    unsigned val;
    Nota *next;
};

struct Estudiante {
    struct{unsigned id; Evaluaciones evaluaciones[4];}val;
    Estudiante *next;
};

struct Curso {
    struct{unsigned id, promocionados, regularizados, recursantes; Estudiante *estudiantes;}val;
    Curso *next;
};

Curso *cursos=nullptr;

2. ProcesarNovedades(nombreDeArchivo, cursos){
    Abrir conexión in al archivo nombreDeArchivo;
    struct {unsigned curso, estudiante, parcial, nota;} s;
    while(LeerRegistro(in,s))
        AgregarNota(cursos, s.curso, s.estudiante, s.parcial, s.nota);
    in.close();
}

3. void AgregarNota(cursos, curso, estudiante, parcial, nota){
    while(cursos no vacía){
        if(c->id==curso){
            while estudiantes no vacía {
                if(e->id==estudiante){
                    while e->evaluaciones[parcial-1] no vacía
                        ;
                    Agregar nota
                    break;
                }
            }
            break;
        }
    }
}

4. void CalcularEstadísiticas(cursos){
    Visitar cada curso, incializar,
        Visitar cada estudiante
            Seleccionar el estado y contar
}

5. unsigned GetEstado(int evaluaciones[]){
    Obtener la última nota de las cuatro evaluaciones
    Si las cuatro son mayores o igual a 8 retornar 1
    Si las cuatro son mayores o igual a 5 retornar 2
    return 3; // recursa
}
```