UTN – FRBA – Algoritmos y Estructura de Datos – Examen Final – 16/02/2017

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, aspectos conceptuales, funciones estructuras de datos.

Contexto

Usted es líder de desarrollo de la Secretaría Académica de una Universidad Nacional. Su 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 deben analizar los datos de los estudiantes inscriptos en una materia de programación.

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

1. Para la regularización se requiere un mínimo de 12 (doce) puntos en la sumatoria de las notas de dos exámenes parciales, o sus correspondientes instancias recuperatorias (dos por cada parcial). Además ninguna de las evaluaciones parciales debe ser menor a 6 (seis)
2. Para estar en condiciones de acceder a la promoción se exige un valor mínimo de 16 (dieciséis) en la suma de las dos evaluaciones parciales. Con un mínimo de 8 (ocho) en cada parcial. Si esto no ocurre y si la nota resulta menor a 8(ocho) pero mayor o igual a 6(seis), el estudiante, no promociona la materia pero logrará la regularidad en el régimen de aprobación por examen final, en la medida que hubiera obtenido calificaciones mayores o iguales a 6(seis) en las instancias de los exámenes parciales. En este caso, quedará comprendido en lo que se establece en el punto anterior.
3. Para poder promocionar definitivamente, además de lo requerido en el punto anterior, se requerirá la aprobación de un tercer parcial integrador (con nota mayor o igual a 8) y la aprobación de un trabajo práctico común con otra asignatura del área de programación también con nota mayor o igual a 8.
4. Si un alumno obtiene una calificación, en la evaluación integradora, mayor o igual a 6 (seis) y menor a ocho y deseara una última oportunidad para aspirar a promocionar, habiendo alcanzado el puntaje necesario para regularizar la asignatura e ir a examen final y con el trabajo practico común e integrador aprobado, el estudiante podrá presentarse a la primera fecha de recuperación con el propósito de alcanzar la promoción.

Definición del problema

Se dispone de la variable cursos que es un vector de colas de N componentes, siendo N un valor conocido a priori que representa la cantidad de cursos, cada cola corresponde a un curso. La estructura enlazada contiene *número de legajo (un dato de tipo entero) y apellido y nombre (una cadena de 20 caracteres) y las evaluaciones de los exámenes* de los estudiantes. Estos datos están ordenados por número de legajo. Se requiere generar un archivo con *legajo*, *nombre y apellido*, y curso de todos los estudiantes de la cátedra que promocionaron en el ciclo lectivo en estudio, ordenado por número de legajo. Los códigos de cursos tienen cuatro dígitos, por ejemplo 1001, 1051 ó 1151.

Restricciones

Debe invocar sin desarrollar las siguientes funciones:

- unsigned GetCurso(unsigned unÍndice); // Dodo unÍndice retorna el código de curso.
- bool IsPromocionado(T exámenes); // Dado los exámenes de un estudiante retorna si promociona o no.
- Dispone además de las funciones de estructuras enlazadas que puede invocar sin desarrollar

Se pide

1. Dé un nombre para T y codifique su declaración para que sea utilizable en la función IsPromocionado. Justifique brevemente.
2. Codifique la declaración de la variable cursos y declare los tipos de datos necesarios para declararla.
3. Codifique el prototipo de **InformarPromocionados** que con el vector de N colas genere el archivo, ordenado por número de legajo (consignando número de legajo, apellido y nombre y curso), de todos los estudiantes que promocionaron en la materia analizada.
4. Codifique o diagrame la función **InformarPromocionados**.

[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. 3 puntos.
3. 1 punto.
4. 4 puntos. // Para aprobar, por lo menos dos puntos tienen que salir del ítem 4.
    a. Itera y usa arreglo correctamente, 1 punto.
    b. Itera y usa cola correctamente, 1 punto.
    c. Itera y usa lista correctamente, 1 punto.
    d. Escribe en archivo correctamente, 1 punto.

Resolución

```
1. Una solución
typedef int Resultados[9]; // tres oportunidades por examen, tres exámenes contando el TP.
// Las notas son enteros en el intervalo cerrado [1, 10], int es un tipo de dato más que suficiente.

1. Otra solución
#define PARCIALES 2
#define TPS 1
#define EXAMENES (PARCIALES+TPS)
#define RECUPERATORIOS_POR_EXAMEN 2
#define INSTANCIAS (EXAMENES*(1+RECUPERATORIOS_POR_EXAMEN))
typedef array<unsigned char,INSTANCIAS> ResultadosDeExámenes;
/* INSTANCIAS es 9 y, en esta resolución, se justifica con la fórmula parametrizada.
Las notas son enteros en el intervalo cerrado [1, 10], por lo que el tipo de dato unsigned char es más que suficiente.
Un poco menos eficiente en cuanto a espacio serían unsigned o int.*/

2.
srtuct Estudiante {int legajo; char nya[20+1]; Resultados resultados;};
srtuct Nodo {Estudiante info; Nodo*sgte;};
struct Curso {Nodo* frente; Nodo* fin;}; // un Curso es una cola de Estudiantes
Curso cursos[N];
array<Curso, N> cursos; // otra forma

3. void InformarPromocionados(string  nombreArchivo, Curso cursos[]); //


4. void InformarPromocionados(string  nombreArchivo, Curso cursos[]){
    struct EstudianteConCurso{int legajo; char[20+1] nya; int curso;} r; // para la lista y el archivo
    struct NodoLista{EstudianteConCurso info; NodoLista* sgte;};
    Estudiante unEstudiante;
    FILE* f = fopen(nombreArchivo,"wb+");
    NodoLista *lista = null;
    for(unsigned i=0; i<N;++i){ // Por cada curso
        r.curso=GetCurso(i); // Convertir el curso con la función dada pasando como parámetro el indice
        while(cursos[i] != null{ // Por cada estudiante, mientras haya datos en la estructura
            unEstudiante = Desencolar(cursos[i].fte, cursos[i].fin)  // Saca un nodo para obtener un estudiante
            if(IsPromocionado(unEstudiante.resultados)){ // Funcion dada Si promociona lo agrego a la lista
                r.legajo = unEstudiante.legajo;// si promociona se deben completar los datos
                r.nya = unEstudiante.nya;
                InsertarOrdenado(lista, r);
            }
        }
    }
    ;
    while(lista!=null){
        r = Sacar(Lista);
        fwrite(&r, sizeof(r), 1,f);
    }
    fclose(f) ;
return;
}
```