UTN – FRBA | Algoritmos y Estructura de Datos | Examen Final 18/12/2019

Apellido y Nombre: | Legajo: | Cursó con Prof.:

Nota:

Temas evaluados: *Abstracción, flujos, estructuras de datos enlazadas, arreglos, resolución de problemas.*

1. La facultad requiere realizar las **Actas de los alumnos regulares** de cada uno de los cursos impartidos en el ciclo lectivo actual. Serán alumnos regulares si NO superan las ocho inasistencias y si además NO aparecen informados en una lista de alumnos dados de baja por cantidad de inasistencias que superaron las ocho. Esta lista en realidad es una estructura dinámica no lineal de tipo árbol binario, la cual contiene: nro. de curso, nro. de Legajo y, fecha de baja (aaaammdd), ordenado por nro. de curso y por nro. de legajo, en ambos casos en forma ascendente. Un alumno puede pertenecer a varios cursos. Un curso contiene varios alumnos.

Se cuenta con las siguientes declaraciones de tipos:

```
typedef char str20[21];
struct sCurso {
   str5  nroCurso;
   int   cantInasist; //Hasta ocho inclusive se considera alumno regular.
   bool  regular; //todos en el estado true.
}
typedef sCurso tvrCur[10];
typedef struct tNodo * tTree;
```

Se cuenta con un archivo binario de datos de **Alumnos.Dat** con el siguiente diseño de registro:

```
struct sAlumno {
     int      nroLeg;
     str20    apellidos,
              nombres;
     int      dni,
              fecNac; // aaaammdd
     tvrCur   vrCursos;
};
```

Se pide:

a) Declare el tipo **tNodo,** del árbol binario de alumnos dados de baja por bedelía. (**1 PUNTO**).

b) Desarrolle, (diagrame o codifique en C++) la siguiente función cuyo prototipo es:

void ActualizarAlumnosRegularesxCursos(**char** nomFisAlumnos[ ], **tTree** &TreeBinBajas);

que establece en **false** el campo vrCursos[i].regular, a cada uno de los alumnos que hayan superado la cantInasist limite o se encuentren informados en la estructura TreeBinBajas. Utilizar la función como caja negra, siguiente:

tTree BuscarBaja(**tTree** &TreeBinBajas, **int** nLeg, **str5** nroCurso);

que recibe una estructura no lineal de alumnos dados de baja por bedelía, busca el nro.de curso y el nro.de legajo de un alumno, en esa estructura, y si NO lo encuentra retorna NULL, caso contrario, retorna un puntero al nodo en donde encontró el nro.de curso y el nro.de legajo del alumno.
Deberá actualizarse en el mismo archivo el nuevo estado de regular si corresponde hacerlo. (**3 PUNTOS**).

c) Desarrolle, (diagrame o codifique en C++) la siguiente función, cuyo prototipo es:

void IncorporarNuevaBaja(? , int nroLeg, **str5** nroCurso, **int** fecBaja);

Que incorpora ordenadamente por nro.de Curso y por nro.de Legajo un alumno que ha sido dado de baja por bedelía. En la ubicación del símbolo ? debe pasarse como parámetro la estructura **tTree** de alumnos dados de baja. En alguna instancia de este proceso se deberá invocacar a la siguiente función, utilizada como caja negra:

void InsertarNodoHoja(**tTree** &TreeBinBaja, **tTree** pNodo);

Establecer el método de pasaje de parámetros adecuado para este caso, justifique la respuesta. (**3 PUNTOS**).

NOTA: Un alumno puede estar cursando varios cursos, máximo 10, pero, si cursa menos estará indicado en el campo nroCurso por un símbolo de asterisco *, que depende del arreglo de registro vrCursos[i].

La nota mínima para la aprobación es 6 (seis). Es condición requerida para la aprobación, además de la nota, que las definiciones de tipos y al menos una función debe estar resuelta correctamente.

UTN – FRBA | Algoritmos y Estructura de Datos | Examen Final 18/12/2019

Apellido y Nombre: | Legajo: | Cursó con Prof.: | Nota:

2

```
short n = 5,
      i = 0,
      j = 2,
      mat[ 19 ] [ 19 ];
for (short num = 1; num <= n * n; num++) {
  mat[ i ] [ j ] = num;
  i--;
  j++;
  if (i == -1 && j == n) {
    i+= 2;
    j--;
  }
  else
    if (i == -1)
      i = n - 1;
    else
      if (j == n)
        j = 0;
      else
        if (mat[ i ] [ j ]) {
          i+= 2;
          j--;
        }
} //fin for
                    // (2 PUNTOS)
```

Dibuje la matriz mat de n filas x n columnas e indique cada uno de los valores que contendrán las celdas.

Comentar si encuentra alguna propiedad o característica una vez generada la matriz con valores enteros.

2.

```
void IndexarAlumnos(ifstream &Alum, fstream &AlumIdx) {
 tListaAlum  Lista = NULL;
 sAlum       rAlum;
 tInfoAlum   rInfo;

 while ( Alum.read((char *) &rAlum,sizeof rAlum) )  {
   rInfo.NroLeg = rAlum.nroLeg;
   rInfo.PosAlum = (Alum.tellg() - (int) sizeof rAlum)
                                / sizeof rAlum;
   InsertaNodo(Lista,rInfo); //incorpora nodo ord. x nroLeg.
 }
 Alum.clear();
 while ( Lista )  {
   SacarPrimerNodo(Lista,rInfo);
   AlumIdx.write((const char *) &rInfo,sizeof rInfo);
   cout << rInfo.NroLeg << "   " << rInfo.PosAlum << endl;
 }
} //IndexarAlumnos     (1 PUNTO)
```

Asumiendo que los nros.de legajos leidos desde el archivo fueran:

37, 21, 89, 14, 42, 99, 4, 19, 76, 55.

Indicar la salida por pantalla.

La nota mínima para la aprobación es 6 (seis). Es condición requerida para la aprobación, además de la nota, que las definiciones de tipos y al menos una función debe estar resuelta correctamente.