Algoritmos y Estructuras de Datos

Ejercicio de Práctica en Papel

Nombre y Apellido: .....

El siguiente ejercicio debe ser resuelto con lapicera/lápiz. Todas las hojas deben tener nombre y apellido. Tener en cuenta la legibilidad de la solución entregada, ya que de no comprenderse lo escrito no se podrá corregir. Deben entregarse todas las hojas que contengan código asociado a la solución. Recuerde que si hace uso de funciones auxiliares, debe incluirlas y/o codificarlas según corresponda.

Tiempo de Resolución: 90 minutos.
Puntaje Requerido: 24/40 puntos.

Consigna: un hacker no benigno (es decir maligno) ha logrado corromper la información del archivo de Alumnado donde están registrados los datos de todos los alumnos, y sólo dejó una matriz de 1450 x 30 caracteres llamada *MatrizRota*, que tiene los datos modificados de los 1450 alumnos activos de la facultad (uno por fila).

En cada fila está almacenado el nombre completo de un estudiante y su año de ingreso a la UTN. En la siguiente figura se muestran algunas filas de *MatrizRota*:

Indice	Contenido Fila[Indice]
0	`E 5v a0r is0 2t o$ P o rt el a`
1	`4 0 0 E ls 2a $ Za m b r an o`
…	…
1449	`J u l i 3 00 2  ana $ Fl ores`

Ud. debe reconstruir la información desde el contenido de cada fila a su formato original:

Portela Evaristo 2005
Zambrano Elsa 2004
…
Flores Juliana 2003

Como ya se habrá dado cuenta cada línea de la matriz tiene todos los caracteres del Nombre y Apellido de cada alumno, de izquierda a derecha, pero con espacios en blanco insertados aleatoriamente. El signo $ es clave para poder separar y distinguir el nombre del apellido. Y los 4 dígitos que aparecen también en posiciones aleatorias de la fila corresponden al año de ingreso (pero vienen invertidos en su orden).

a) Ud. debe definir las estructuras de datos necesarias y codificar la función *ReconstruyeLinea()* que retornará una estructura de datos *Alumno* con la información original de un alumno. La función *ReconstruyeLinea( )* recibirá la información de una fila de la matriz, recuperará su versión original y la retornará. (NO se debe modificar *MatrizRota).*

b) Además debe codificar la función *ReconstruyeInfo()* que recibirá la *MatrizRota* y deberá retornar: la información original de la matriz en un dato del tipo *ListaAlumnos* (hay filas de *MatrizRota* que no se pueden recuperar, contienen la subcadena “DANGER” y no deberán retornarse en el vector reconstruido). El tipo *ListaAlumnos* queda definido como:

```
struct ListaAlumnos {
      Alumno lista[1450];
      int tl;
}
```

c) El siguiente código:

```
void itemC(int v[], const int tope) {
      int i, k, x;

      for (i=1; i<tope; i++){
            x = v[i];
            k = i-1;
            while ( (k>=0) && (x<v[k]) ) {
                  v[k+1] = v[k];
                  k--;
                  }
            v[k+1]= x;
      }
}
```

corresponde a un algoritmo de ordenamiento para vectores con elementos enteros:
c1) Identifique a qué algoritmo corresponde y explique brevemente su idea de funcionamiento.
c2) Modifique/reescriba el código de la función, para que ordene el struct *ListaAlumnos* obtenido en el ítem b, ascendentemente según año de ingreso de los alumnos.

d) Se ha procesado la información de los alumnos que ingresaron en los años 2000, 2001 y 2002 y se dispone de un dato *ListaAlumnos* para cada uno de esos años. Dichas listas (llamadas L2000, L2001 y L2002) están ordenadas alfabéticamente por Apellido y Nombre. Ud. debe codificar la función *Fusion()* que reciba dos listas de Alumnos y las fusione en una tercera que contendrá el listado completo de alumnos de las dos listas, ordenado alfabéticamente.
Codifique las llamadas necesarias a Fusion() para poder fusionar las tres listas, y tener la información de todos los alumnos en la lista LTotal.

Importante: Para la resolución del problema el alumno puede codificar todas las estructuras de datos y funciones que considere necesarias. Los campos de las estructuras deben respetar lo enunciado en la consigna. En los casos en los que no se indica un prototipo, los parámetros formales de las funciones (cantidad y tipo) deben definirse según los objetivos propuestos. El puntaje final obtenido tendrá en cuenta la eficiencia de la estrategia de resolución elegida.