TAREA Nº3

Programación en C: Aplicación de Árboles Binarios.

Fecha de envío: lunes 26 de junio, a las 23:59 hrs.
Modalidad: Trabajo individual

I. OBJETIVOS.

El objetivo de la presente tarea es evaluar tu capacidad para:

1. Llevar a cabo un programa completo en el lenguaje de programación C, utilizando todos los elementos básicos que provee el lenguaje, y sus tipos de datos básicos.
2. Aplicar las buenas prácticas en programación (orden, comentarios, identificadores representativos).
3. Ser capaz de cumplir con una interfaz concreta solicitada para el programa.
4. Revolver un problema que requiera búsqueda y ordenamiento de información aplicando árboles.

II. ENUNCIADO.

En esta tarea utilizarás el mismo contexto que en la tarea 1, pero en esta ocasión deberás utilizar un **árbol binario ordenado** para determinar las personas que están intentando comprar más de las dos entradas que se permitían en la promoción. Para lograr lo anterior, tu árbol debe tener como clave (de ordenamiento) el rut de la persona.

Adicionalmente, en cada uno de los nodos del árbol deberás almacenar el nombre y la cantidad de entradas que la persona desea comprar. Debes registrar una única vez en el árbol cada uno de los ruts que estén en el archivo de entrada. La primera vez que se crea el nodo, el número de entradas es el que aparece en la línea del archivo con que se creó el registro. Si una persona aparece más de una vez en el archivo de entrada, la segunda vez (y las siguientes veces) debes buscar su registro en el árbol usando el rut como clave y sumarle la cantidad de entradas adicionales que la persona está intentando comprar. Como resultado final tu programa debe generar un archivo que indique las personas que intentaron adquirir más de las dos entradas permitidas en la promoción indicando la cantidad total solicitada.

Se pide:

Debes construir un programa en C, que cumpla con los siguientes pasos:

1. Solicitar el archivo de entrada al usuario.

   El formato que poseerá será el mismo que en la tarea 1. Cada una de las líneas del archivo de entrada contendrá cada inscripción realizada en la página web con la promoción y su formato será:

   ```
   rut,nombre_apellido,número_de_entradas_solicitadas
   ```

2. Si hay personas que solicitaron en total más de 2 entradas, registrarlas en un archivo de salida de extensión "**.sde**" ("sobre dos entradas").

   El archivo de salida debe tener el mismo nombre que el de entrada pero con la extensión cambiada a "**.sde**". El usuario debe ingresar nombre y extensión (No asumas que siempre tendrá extensión ".txt").

   Si todas las personas del archivo de entrada solicitaron 1 ó 2 entradas, el archivo de salida no se debe generar.

   Si se genera el archivo de salida, las personas ingresadas deben aparecer ordenadas por rut de menor a mayor, siguiendo este formato:

   ```
   rut, nombre_apellido: total_de_entradas
   ```

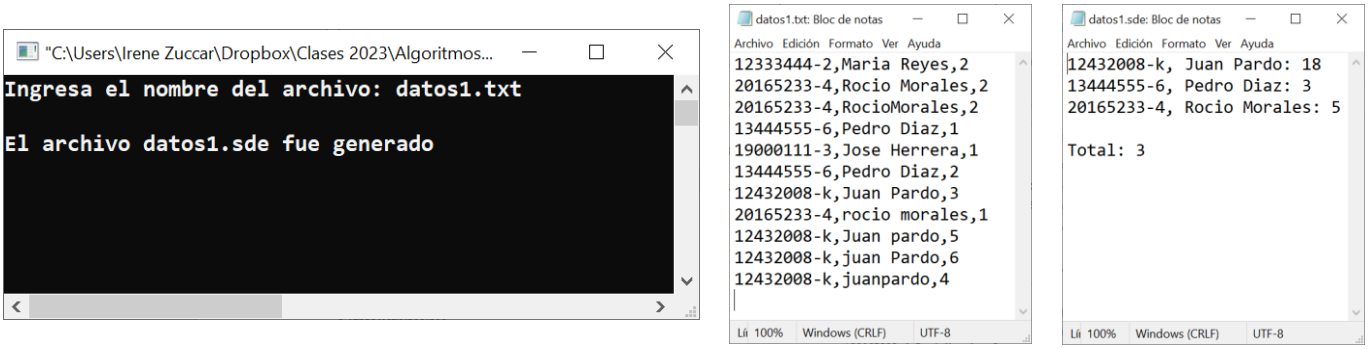
   Al final del archivo debe aparecer el total de personas que posee el archivo.

3. Debe haber un mensaje final al usuario:

   Si el archivo se genera: "**`El archivo <nombre del archivo de salida> fue generado`**"
   Si el archivo no se genera: "**`Todos solicitaron la cantidad correcta de entradas`**"

Ejemplos:

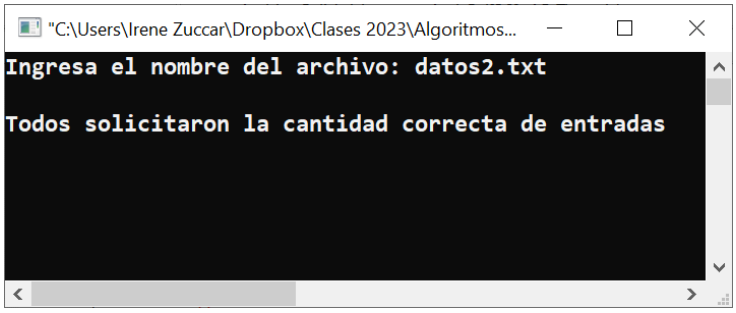
En la figura 1 aparece un ejemplo del caso en el que se debe generar un archivo de salida. En la figura 1(a) aparece la interfaz solicitada; en la figura 1(b) se muestra el contenido del archivo de entrada y en la figura 1(c) se muestra el contenido que debe tener el archivo de salida correspondiente: observa que los ruts están ordenados de menor a mayor.



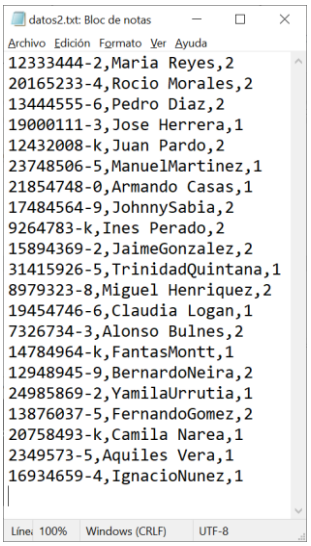
(a) **(b)** **(c)**

Figura 1: Ejemplo de lo que debe obtener tu programa: **(a)** Interfaz; **(b)** Ejemplo de archivo de entrada; **(c)** Archivo de salida generado.

En la figura 2 aparece un ejemplo del caso en el que **NO** se debe generar un archivo de salida. En la figura 2(a) aparece la interfaz solicitada; en la figura 2(b) se muestra el contenido del archivo de entrada.



(a)



(b)

Figura 2: Ejemplo de lo que debe obtener tu programa: **(a)** Interfaz; **(b)** Ejemplo de archivo de entrada.

III. CONSIDERACIONES EN LA REVISIÓN.

Sobre el código:

1. Debes resolver esta tarea usando árboles binarios ordenados a través de su representación computacional dinámica. No usar esta representación significa tener una nota **1.0** en esta tarea.

2. En esta tarea **no podrás usar variables globales**.

3. Tu programa **debe seguir exactamente** las **reglas e interfaces** explicadas en este documento.

4. Debes construir **una función para cada tarea que realice tu código**, cuidando los tipos de datos de las entradas y de las salidas.

5. Debes usar identificadores representativos para tus variables, parámetros de entrada y funciones.

6. Tu código debe estar correctamente *indentado* (uso de sangrías para cada sub-bloque de instrucciones), esto incluye el correcto alineamiento de las llaves (“**{**” y “**}**”) que enmarcan tales bloques.

7. Tu código no puede presentar más de 1 línea en blanco. Se penalizarán códigos que no cumplan con este punto.

8. El sistema debe ser **robusto**, se penalizarán los errores no manejados de cualquier tipo.

9. Debes comentar cada una de tus funciones, estructuras y tipos de datos que definas, indicando una descripción de la labor que lleva a cabo, sus entradas, y sus salidas (ver ejemplo en figura 3). Estos comentarios deben estar arriba de lo comentado.

```
/*
Entrada: número entero que se revisará.
Salida: número entero que corresponde al número de divisores.
Proceso: función que recibe un entero y calcula y retorna el total
         de divisores que posee.
*/
int numeroDivisores(int n)
{
    int i=2, cont=0;

    while (i < n)
    {
        if (n%i == 0)
        {
            cont++;
        }
        i++;
    }
    return cont+2;
}
```

Figura 3: Ejemplo de cómo debes comentar una función en esta tarea.

10. Puedes trabajar con el IDE y compilador de C que más te acomode. No obstante, tu código debe poder ser compilado y ejecutado sin problemas en Windows.

IV. SOBRE LA ENTREGA, ATRASOS Y FALTAS A LA ÉTICA.

1. Debes subir tu trabajo **al aula virtual de tu curso de cátedra** dentro de la plataforma **https://canvas.unab.com/,** en una casilla que se habilitará especialmente para esto.

2. **NO SE ACEPTARÁN TRABAJOS ATRASADOS.**

3. Tu trabajo es **individual**, no se aceptarán trabajos en grupo.

4. Debe subir **solo un archivo** con el código fuente de tu programa.

5. El nombre de tu archivo debe ser "**paterno_materno_nombre.c**"

6. Si el programa no se puede ejecutar, tendrá la nota mínima: **1.0**. Si tu programa funciona, se evaluará su ejecución, y también el código fuente.

7. Ante el escenario de existir sospecha de copia (con otros compañeros, desde internet o realizado por una persona distinta a quien entregó), será interrogado acerca de su trabajo, para aclarar dudas de su entendimiento y autoría. Si se confirma la sospecha, el trabajo será evaluado con nota **1.0**.

8. Las consultas las debe realizar directamente a los profesores de taller, presencialmente en clases o a los correos y en los horarios que ellos establezcan.