TAREA Nº1

Programación en C: Aplicación de Listas Dinámicas.

Fecha de envío: sábado 17 de septiembre, a las 23:55 hrs.
Modalidad: Trabajo en grupos de a lo más **dos** personas

I. OBJETIVOS.

El objetivo del presente laboratorio 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. Diseñar e implementar una estructura lineal dinámica para resolver un problema que requiera inserción y análisis de información.

II. ENUNCIADO.

Un ADN (ácido desoxirribonucleico) es una molécula que contiene las instrucciones utilizadas en el crecimiento, desarrollo, funcionamiento y reproducción de todos los organismos vivos y algunos virus. Una cadena de ADN está compuesta por bases nitrogenadas denominadas adenina(**a**), citosina(**c**), timina(**t**) y guanina(**g**).

El Dr. Carlos Javier tiene un gran problema, el cual es identificar a posible mutantes para su escuela de entrenamiento. La idea es construir una máquina que pueda leer la cadena de ADN y comprobar si la persona es mutante o no. Después de mucha investigación, el Dr. Javier con sus colaboradores, la Dra. Yein y el Dr. Beest han determinado que las siguientes características están presentes en una cadena de ADN mutante:

- La cadena no puede contener tres guaninas consecutivas.
- La cadena debe tener dos adeninas o dos citosinas consecutivas.
- Todas las cadenas mutantes, terminan con una timina.

Se requiere que ayudes al Dr. Javier a detectar a los posibles mutantes, procesando cadenas de ADN.

Se pide:

Debes construir un programa en C, que reciba el nombre de un archivo de texto de entrada, con una secuencia de caracteres, y que muestre por pantalla “Humano” o “Mutante” según el análisis de tu programa. Respetar el orden de inscripción que aparece en el archivo de entrada. En la figura 1 verás un ejemplo de la interfaz.

```
Ingresa el nombre del archivo: datos1.txt
Mutante
```

Figura 1: Ejemplo de la interfaz que debe tener tu programa. En este ejemplo el archivo de entrada se llama "datos1.txt" y debe ser ingresado por el usuario.

III. CONSIDERACIONES EN LA REVISIÓN.

Sobre el código:

1. **Debes usar listas dinámicas para resolver esta tarea. Si se ocupa otro tipo de solución, la nota será 1.0.**

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

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

4. Debes usar nombres representativos para tus variables, parámetros de entrada y funciones.

5. 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. En la figura 2 aparece un ejemplo de cómo debes hacerlo.

```
/*
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 2: Ejemplo de cómo debes comentar una función en esta tarea.

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. Se penalizarán códigos desordenados en este aspecto.

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

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

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

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://unab.blackboard.com/,** en una casilla que se habilitará especialmente para esto.

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

3. Tu trabajo debe ser subido por **SOLO UNO de los integrantes del grupo**.

4. Debes subir solo el código fuente.

5. El nombre del código fuente debe ser "**rutcompleto1_rutcompleto2.c**", indicando el rut de cada integrante del grupo. Por ejemplo, si tu rut es 19.000.111-3, debieras ingresarlo sin puntos ni guión, pero sí con el dígito verificador: o sea 190001113.

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

7. Ante el escenario de existir sospecha de copia (con otros compañeros, o desde internet), 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.