TRABAJO PRÁCTICO FINAL

Pautas generales:

- El siguiente **Trabajo Práctico** puede ser realizado individualmente o en grupo con hasta 3(tres) integrantes máximo (sin excepción).

- Deberá informarse al docente de la comisión cómo está conformado su equipo de trabajo vía mail (uno solo por equipo), pueden establecer un nombre para identificarlos fácilmente. Es necesario que, una vez establecido e informado quienes son los integrantes del grupo, no se soliciten cambios respecto a ello.

- Considerar que todos los integrantes deben conocer todos los aspectos del trabajo entregado.

- Se admite una única entrega final, es por ello que solicitamos revisen muy bien las funcionalidades previamente a la entrega formal, que debe comunicarse con un mail al docente de la comisión, en ese mail deben enviar la documentación pertinente y acceso al repositorio git.

- Pueden consultar a los docentes en caso de dudas o bien para ir certificando correctitud durante el proceso de implementación.

- Se recomienda, una vez comunicados los detalles sobre cómo estará conformado el equipo de trabajo, avanzar con la implementación de los bash scripts.

- Se comentará en clases material e instrucciones sobre las tareas relacionadas a git y docker.

<u>Aspectos evaluativos:</u>

- programación bash
- uso de git / gitlab o github
- uso de contenedores
- generación de documentación

<u>Requerimientos generales:</u>

- programar en **bash** una serie de scripts para análisis de texto
- utilizar un repositorio para desarrollo mediante sistema de control de versiones **git**
- permitir el acceso al repositorio git a los docentes vía token
- generar un **Dockerfile** para ejecutar la aplicación
- generar documentación de todos los elementos que intervienen en el trabajo práctico
- mostrar cómo ejecutan la app desde cero, desde la ejecución del contenedor hasta el uso de la misma

Entregables:

- Documento con los detalles solicitados en el apartado anterior
- Identificación del repositorio en gitlab/github, acceso vía token
- Demostración en vivo de ejecución (esto se hace ante los docentes en mesa final o fecha previa que se acuerde conjuntamente, puede ser presencial o virtual, según se establezca)

Comentarios Finales:

- Todos los participantes del grupo deben estar al tanto de cada aspecto y detalle del trabajo, asegurarse de ello pues serán consultados por cada característica, indistintamente sea el integrante del equipo.

OBJETIVOS

El trabajo consiste en generar un contenedor que al ejecutarse presente un menú de opciones de filtrados de un texto dado. Deberán editar un Dockerfile y construir una imagen.

El texto a analizar debe ser depositado en un directorio del equipo host y copiar el contenido del mismo al contenedor para que lo tenga disponible para su posterior análisis.

Tanto los scripts, como el Dockerfile y el archivo de texto de entrada, deben estar en el repositorio gitlab o github que creen para realizar el desarrollo en equipo.

Recomendamos trabajar en el repositorio manteniendo la prolijidad y las buenas prácticas de git. Una estrategia recomendable es realizar un branch por cada ejercicio e ir integrando a la rama principal a medida que los scripts que resuelven cada ejercicio estén listos.

En ese mismo repositorio debe estar la documentación suficiente para comprender cómo desplegar el contenedor y ejecutar la aplicación, con algunas capturas de ejecución o lo que consideren logre explicitar el funcionamiento y visualizar resultados. Puede ser utilizando un archivo README.md el cual estará presente cuando se accede al repositorio en gitlab o github.

Funcionalidades

Menú de mini aplicaciones (scripts de bash) para análisis de texto.
Dependiendo de la cantidad de integrantes, deberá cumplirse con la implementación de:

- 1 Integrante - ítems del 1 al 5
- 2 integrantes - ítems del 1 al 9
- 3 integrantes - ítems del 1 al 14

1. **statsWords.sh**
   Indicador estadístico de longitud de palabras (la más corta, la más larga y el promedio de longitud).

   WC
2. **statsUsageWords.sh**
   Indicador estadístico de uso de palabras, deben ser de al menos 4(cuatro) letras. (mostrar un Top Ten de estas palabras ordenadas desde la que tiene más apariciones a la que tiene menos). Es decir, filtrar las palabras que tengan al menos 4 letras y de éstas, elegir las 10 más usadas.

3. **findNames.sh**
   Identificación de nombres propios (se identifican sólo si están en este formato *Nnnnnnnn*), aunque la palabra no sea un nombre propio realmente.
   Ejemplos: Mateo, Estonoesunnombre, Ana.

4. **statsSentences.sh**
   Indicador estadístico de longitud de oraciones (la más corta, la más larga y el promedio de longitud).

5. **blankLinesCounter.sh**
   Contador de líneas en blanco.

6. **caseConverter.sh** con el sed convertir las a a A y asi
   Invertir minúsculas a mayúsculas, y viceversa, de todas las palabras.

7. **substringReplace.sh**
   Reemplazo de subcadenas, que considere diferencias entre minúsculas y mayúsculas, pero ignore acentos. Recibe dos cadenas, y cada aparición de *cadena1* debe reemplazarse por la *cadena2*.
   Ejemplo: cadena1: tre cadena2: TRE
   las palabras: enTrepiso, entretenido, intrépido
   pasan a: enTrepiso, enTREtenido, inTREpido

8. **blockSelection.sh**
   Selección de oración y/o párrafo en base a un número de entrada. Es decir, se puede establecer como entrada “O” o “P” para indicar oración o párrafo y luego un número (un párrafo se distingue de otro con un punto y aparte, las oraciones vía un punto seguido).

9. **palindromeDetection.sh**
   Mostrar palabras palíndromo (ignorar mayúsculas/minúsculas y acentos en este caso). Ejemplos: Neuquén, radar, reconocer

10. **oneVowelWords.sh**
    Detectar palabras que tengan una sola vocal diferente y más de tres letras (ejemplos: yarará, menesteres, cómodo) , mostrarlas lexicográficamente, es decir, siguiendo el orden en el que aparecen en el diccionario. (sin repetir e ignorar letras acentuadas).

11. **allUpperCase.sh**
    Listar lexicográficamente las palabras que se presenten con todas sus letras en mayúsculas (no repetir palabras en el listado).
    Ejemplos: EJEMPLO, CANCIÓN

12. **allVowelsInWord.sh**
    Mostrar palabras que tengan todas las vocales incluidas al menos una vez (*ejemplos: murciélago y sexagésimocuarto*), mostralas en orden alfabético, si hay más de una aparición colocar entre paréntesis al lado de la misma dicha cantidad de apariciones.

13. **mailAddressDetection.sh**
    Detectar y listar las direcciones de mail encontradas dentro del texto, ordenarlas y evitar repeticiones. Ejemplo: nombre@dominio.com
    Ayuda: utilizar una expresión regular.

14. **integerDetection.sh**
    Detectar números enteros dentro del texto y mostrarlos de menor a mayor, evitando repeticiones. Ejemplo: 123 o 22065
    Ayuda: utilizar una expresión regular.
    Reducción de complejidad: no considerar números reales o en otra notación, por ejemplo: 33,2 o 13,4 en estos casos tratar cada parte como un número independiente.

Toda duda o comentario dirigirla a los docentes de cada comisión.