INF-134 Estructuras de Datos, 2019-2

Tarea 1

Profesor: Diego Arroyuelo B.
darroyue@inf.utfsm.cl

Ayudantes:
Gabriel Carmona gabriel.carmonat@sansano.usm.cl,
Hector Larrañaga hector.larranaga@sansano.usm.cl.

Fecha de entrega: 27 de septiembre, 2019
Plazo máximo de entrega: 5 días.

1. Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorios B-032. Debe usarse el lenguaje de programación C. Al evaluarlas, las tareas serán compiladas en el terminal de linux usando `gcc archivo.c -o output -Wall`. Recordar que una única tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobar el curso.

2. Objetivos

Comprender y familiarizarse con las estructuras y tipos de datos básicos que provee el Lenguaje de Programación C. Entre los conceptos mas importantes, se encuentran:

- Paso de parámetros por valor.
- Paso de parámetros por referencia.
- Asignación de memoria dinámica.
- Manipulación de punteros.
- Manejo de E/S.
- Recursividad

Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

3. Problemas a Resolver

En esta sección se requiere que implementen varias funciones en C. Cada una de éstas debe estar en archivos `.c` separados en su entregable, con la correspondiente función `main`.

Problema 1: Prefijo Común Más Largo

Dado un string $z[1 \dots n]$, se define como prefijo de z a todo string $z[1 \dots i]$, tal que $i = 1, \dots, n$. Por ejemplo, para $z[1 \dots 11]$ = `mississippi`, tenemos 11 prefijos posibles: `m`, `mi`, `mis`, `miss`, `missi`, `missis`, `mississ`, `mississi`, `mississip`, `mississipp`, y `mississippi`.

Dados dos strings z y w, el prefijo común mas largo entre ellos es el string v más largo tal que v es prefijo tanto de z como de w. Por ejemplo, dados los strings `mississippi` y `miseria`, el prefijo común más largo entre ambos es `mis`, de largo 3.

Una aplicación para detectar plagios en textos necesita la siguiente funcionalidad. Dado un conjunto de strings, se quiere encontrar aquel par de strings del conjunto que tenga el prefijo común más largo. Implementar la función `int pCML (char **S, int n)`, la cual recibe como parámetro un conjunto `S` de `n` strings, y devuelve el mayor entero x tal que dos strings z y w en `S` tienen prefijo común más largo de longitud x. Por ejemplo, dado el conjunto de strings `flor`, `hola`, `fracción`, `flotante`, la función debe retornar 3, el cual corresponde al prefijo `flo` entre los strings `flor` y `flotante`.

Dado que la función se usará para procesar conjuntos grandes de strings (hasta 10 millones de strings), es importante que su solución sea eficiente. Una solución ineficiente en cuanto a tiempo de ejecución (por ejemplo, una que chequee todo los posibles pares de strings para buscar el máximo) podría ser evaluada con el 50 % de los puntos. Para la implementación, se pueden usar funciones de manipulación de strings y/o alguna otra función necesaria de la librería standard de C.

Bonus: se dará 20 puntos de bonus a los tres grupos que logren el menor tiempo de ejecución total (y además respondan correctamente).

Formato de Entrada

La entrada de datos será a través del archivo '`strings.txt`', el cual contiene un conjunto de strings a ser procesados. El archivo tiene un string por línea, y es terminado por `EOF`. La longitud máxima de cada string en el archivo es 200 chars, y puede haber hasta 10 millones de strings en un archivo.

Por ejemplo:

```
flor
hola
fracción
flotante
```

Formato de Salida

Debe imprimirse en el archivo '`salida-1.txt`' el resultado de invocar a la función `pCML` con los datos provistos en el archivo de entrada.

La salida correspondiente al ejemplo anterior es la siguiente:

```
3
```

Problema 2: Descomprimir Strings

Implemente la función `char * descomprimir (char *nbreArchivo)`, en lenguaje C, la cual recibe como parámetro un string `nbreArchivo` con un nombre de archivo, y retorna el string resultante de descomprimir el contenido del archivo `nbreArchivo`.

Suponga que el archivo cuyo nombre es provisto como parámetro es binario y tiene el siguiente formato:

- El primer elemento del archivo es un `int` de 4 bytes (llamemos `n` a dicho entero).
- A continuación le siguen `n` valores de tipo `unsigned int` de 4 bytes cada uno. Llamemos `Repeticiones` al arreglo que contiene dichos valores.

- Finalmente, el archivo contiene `n` valores de tipo `unsigned char`, de 1 byte cada uno. Llamemos `Letras` al arreglo que contiene dichos valores.

El string descomprimido (y retornado por la función) contiene los mismos caracteres que `Letras`, y en el mismo orden. La diferencia es que en el string resultante, cada caracter `Letras[i]` aparece `Repeticiones[i]` veces consecutivas. Por ejemplo, si `n=5` y tenemos

```
Letras[5] = {'a','g','t','a','f'}
```

y además

```
Repeticiones[5] = {3,1,5,3,2}
```

entonces se debe generar el string de salida "`aaagttttaaaff`".

Importante: para generar el string descomprimido, se debe usar la cantidad exacta necesaria de memoria. De no cumplirse esto, habrán importantes descuentos en la nota.

Formato de Entrada

Se leerá un archivo llamado '`input-strings.txt`' con los strings que indiquen los nombres de archivo a descomprimir. Cada nombre de archivo tiene a lo más 100 caracteres de largo. La cantidad de nombres de archivo en la entrada es indeterminada, siendo el archivo terminado por `EOF`.

Formato de Salida

Por cada nombre de archivo en el archivo de entrada, el programa debe invocar a la función `descomprimir`, y el string resultante debe imprimirse en el archivo '`strings-descomprimidos.txt`', con un string por línea. El archivo de salida debe tener tantas líneas como el archivo de entrada. En caso de que un archivo no exista, se debe imprimir "`Archivo no existe`" (sin las comillas) en la línea correspondiente.

4. Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea1-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda) a la página aula.usm del curso, a más tardar el día viernes 27 de septiembre de 2019, a las 23:59:59 hs (Chile Continental), el cual contenga:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. ¡Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, Paralelo y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias.

5. Restricciones y Consideraciones

- Por cada día de atraso en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.
- Las tareas deben compilar en los computadores que se encuentran en el laboratorio B-032. **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Por cada *Warning* en la compilación se descontarán 5 puntos.

- Si se detecta **COPIA** la nota automáticamente sera 0 (CERO), para todos los grupos involucrados. El incidente será reportado al jefe de carrera.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

6. Consejos de Programación

El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Por cada comentario faltante, se restarán 5 puntos.

Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por **cada bloque mal indentado, se quitarán 10 puntos**.