INF-134 Estructuras de Datos, 2018-1

Tarea 1

Profesores: Diego Arroyuelo, Natalia González, Roberto Díaz
darroyue@inf.utfsm.cl, natalia.gonzalezg@usm.cl, robertodiazurra@gmail.com

Ayudantes:
Camilo Saldias (camilo.saldias.12@sansano.usm.cl)
Abdel Sandoval (abdel.sandoval@sansano.usm.cl)
Ignacio Tampe (ignacio.tampe@sansano.usm.cl)
Alejandro Vilches (alejandro.vilches@sansano.usm.cl).

Fecha de entrega: 27 de Abril, 2018
Plazo máximo de entrega (atrasos): 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 LDS – B-032 (Fedora 17 de 64 bits). Debe usarse el lenguaje de programación C. Se recomienda compilar en el terminal 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.

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`.

3.1. Problema 1: Combinar Archivos

Se requiere la implementación de un programa que sea capaz de combinar archivos. Para ello se asumirá el caso de una universidad similar a la UTFSM, que almacena la información de sus alumnos en archivos de distintos formatos.

Asuma las siguientes definiciones para `curso` y `nota`:

```
typedef struct {
  char sigla[7];
  int semestre;  /* 1er o 2do semestre*/
} curso;

typedef struct {
  char rolEstudiante[12];
  char siglaCurso[7];
  int nota;
} nota;
```

El programa a implementar deberá leer desde varios archivos con distintos formatos los datos de estudiantes, cursos y notas, y entregar en archivos separados los alumnos que aprobaron todos los ramos del primer semestre, y los que aprobaron los del segundo semestre.

Formato de Entrada

La entrada de datos será a través de tres archivos: `alumnos.txt`, `cursos.dat`, y `notas.dat`. El archivo `alumnos.txt` tendrá formato ASCII, con la siguiente estructura: en la primera línea un único número entero `N`, al que le siguen `N` líneas, cada una correspondiente a un alumno con su Rol, primer nombre, primer apellido, y año de ingreso. Todos estos valores dentro de la línea estarán separados entre sí por un único espacio. Un ejemplo del archivo `alumnos.txt` es el siguiente:

```
4
201873502-6 Camilo Sandoval 2018
201873503-8 Abdel Tampe 2018
201973554-7 Ignacio Vilches 2019
201673511-2 Alejandro Saldias 2016
```

El archivo `cursos.dat` estará en formato binario. Contendrá primero un entero `C` seguido por `C` cursos en formato binario[1]. El archivo `notas.dat` estará en formato binario. Contendrá primero un entero `M` seguido por `M` notas en formato binario[2].

Formato de Salida

La salida debe imprimirse en los archivos `aprobados-s1.txt` y `aprobados-s2.txt`. El primer archivo contiene los alumnos que aprobaron con nota ≥ 55 todos los ramos del primer semestre, y el segundo los alumnos que aprobaron todos los ramos del segundo semestre. El formato de estos archivos debe ser ASCII y debe contener en cada línea, separados por espacios, el nombre, apellido y rol del alumno, seguido por cada curso, la sigla del curso y la nota que obtuvo. Un ejemplo para el formato de estos archivos se da a continuación:

```
Camilo Sandoval 201873502-6 MAT021 87 IWI131 87
Abdel Tampe 201873503-8 MAT021 100 IWI131 56
```

[1] Recuerde el tipo `curso` definido anteriormente.
[2] Recuerde el tipo `nota` definido anteriormente.

Consideraciones

Se proveerán archivos de prueba de entrada para `alumnos.txt`, `cursos.dat`, y `notas.dat`, a través de las plataformas Moodle/Aula. Sin embargo, se recomienda generar y probar con otros archivos el correcto funcionamiento del programa.

Problema 2: Búsqueda en la Web

Una importante aplicación de búsqueda en la web necesita responder de forma eficiente las consultas de sus usuarios. El objetivo de esta parte de la tarea es procesar los datos de entrada que serán provistos, para lograr tiempos de respuesta acordes con las expectativas de un usuario típico. Para simplificar, la aplicación sólo permite hacer consultas de dos palabras, entregando como resultado todas las páginas que contienen esas dos palabras. Su tarea será, además, ejecutar una serie de consultas que serán entregadas como entrada.

Formato de Entrada

La entrada de datos será a través de los archivos `palabras.dat` y `consultas.dat`. El primer archivo tiene formato ASCII, e indica en qué página web aparece cada palabra, cada línea correspondiendo a una palabra. Esto es, la primera línea corresponde a la palabra `0`, la segunda línea corresponde a la palabra `1`, y así sucesivamente. El archivo tiene un número no determinado (a priori) de líneas, y está terminado por `EOF`. El formato de cada línea es el siguiente: comienza con un valor entero `N` > 0, y le siguen `N` enteros que indican en qué paginas web aparece la palabra en cuestión. Todos los valores dentro de cada línea están separados por un único espacio. Un ejemplo de archivo `palabras.dat` es el siguiente:

```
5 9 1 3 12 4
3 2 5 9
7 6 8 1 9 4 10 11
1 7
```

el cual contiene información de 4 palabras. La primera palabra (palabra `0`), aparece en las 5 páginas web identificadas con los valores enteros 9, 1, 3, 12, y 4. La palabra `1` aparece en las páginas web 2, 5, y 9, y así siguiendo.

El archivo `consultas.dat`, por otro lado, tiene formato ASCII, y contiene las consultas que serán realizadas sobre los datos antes procesados. El archivo contiene un número no determinado de líneas, y finaliza con `EOF`.Cada línea del archivo corresponde a una consulta, y por lo tanto contiene dos valores enteros (separados por un único espacio) que corresponden a las palabras que se van a consultar. Un ejemplo (relacionado al ejemplo anterior) es el siguiente:

```
0 1
2 1
2 0
1 3
2 3
```

La primera consulta es por las palabras `0` y `1`, la segunda por las palabras `2` y `1`, y así siguiendo.

Formato de Salida

Su programa debe procesar los datos del archivo `palabras.dat` y luego producir la respuesta a todas las consultas del archivo `consultas.dat`. La salida debe hacerse en el archivo `salida.dat`, el cual tiene una línea por cada consulta hecha. Cada línea i tiene el siguiente formato: comienza con un entero `M`, y le siguen `M` enteros que corresponden a las páginas que contienen ambas palabras de la línea i del archivo `consultas.dat`. Para facilitar la evaluación y comparación de resultados, los `M` valores enteros dentro de cada línea deben estar ordenados de forma creciente. El archivo resultante para el ejemplo presentado anteriormente es el siguiente:

```
1 9
1 9
3 1 4 9
0
0
```

Esto es, la única página que contiene las palabras `0` y `1` es la 9. A su vez, también la única página que contiene las palabras `2` y `1` es la 9. Por otra parte, las páginas 1, 4, y 9 son las que contienen a ambas palabras `2` y `0`. Finalmente ninguna página contiene a la vez las palabras `1` y `3`. Lo mismo sucede para las palabras `2` y `3`.

Consideraciones

Se recomienda procesar los datos del archivo `palabras.dat`, de tal manera que la respuesta a las consultas sea rápida. Es tarea de cada grupo desarrollar una manera de procesar los datos. Para el procesamiento se permite usar algoritmos adicionales de las librerías standard de `C`. Las 5 tareas que resuelvan este problema con menor tiempo de ejecución, recibirán un bonus de 15 puntos. Dicho bonus podrá ser usado en cualquier otra tarea del curso durante el semestre 2018-1. Aquellos grupos que resuelvan el problema de forma notoriamente lenta, pueden llegar a recibir descuentos.

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) al moodle del curso, a más tardar el día viernes 27 de abril de 2018, a las 23:55:00 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 los laboratorios LDS (Fedora 17 de 64 bits). **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**.