

Trabajo Práctico Especial Programación Imperativa Julio 2023

1. Objetivo

Diseñar e implementar un programa para el **procesamiento de datos de alquileres de bicicletas**, basado en datos reales. Para ello se deberá realizar tanto el *front-end* como el *back-end*, este último **basado en la creación de al menos un TAD**.

Para este trabajo se busca poder procesar los datos de dos ciudades: **Montreal, Canadá** 🇨🇦 y **Nueva York, Estados Unidos** 🇺🇸. Ambos datos son extraídos de los respectivos portales de las empresas concesionarias del sistema de alquiler de bicicletas de la ciudad en formato CSV ([Open Data | BIXI Montréal](#) y [Citi Bike System Data - NYC](#)).



2. Descripción funcional

El programa consiste en dos ejecutables distintos, uno para cada dataset, donde cada uno procese un conjunto de *queries* (consultas) sobre los datos de los viajes en bicicleta.

Cada uno de estos ejecutables deberá leer dos archivos CSV: uno con la información de los **alquileres de bicicletas** correspondientes a una ciudad y otro archivo con la información de las **estaciones de alquiler de bicicletas** de esa ciudad. En ambos el delimitador es un punto y coma (",").

Los datasets a utilizar se encuentran en [Datasets Alumnos](#) y contienen la información filtrada (se eliminaron algunas columnas irrelevantes para el trabajo práctico especial).

A continuación se listan los dos ejemplos de uso que se busca para la aplicación: procesar los datos de alquileres de bicicletas de Montreal 🇨🇦 y de Nueva York 🇺🇸. Sin embargo, es importante recordar que la mayor parte de la implementación no debe estar atada a la realidad de los ejemplos de uso. **Por ejemplo, las estaciones serán las que la aplicación obtenga en ejecución a partir del archivo de estaciones y no serán aceptadas implementaciones que tengan fijos estos datos.** En otras palabras, la implementación deberá funcionar también para procesar los datos de

alquileres de bicicletas de cualquier otra ciudad, manteniendo siempre la estructura de los archivos que se presentan a continuación. Si bien se mencionó más arriba el origen de los archivos, los mismos fueron modificados para simplificar su procesamiento.

Los archivos tienen las siguientes características:

Datos de alquileres de Montreal , a partir de ahora **bikesMON.csv**

El archivo CSV tiene la siguiente estructura:

- **start_date**: Fecha y hora del alquiler de la bicicleta (inicio del viaje) en formato yyyy-MM-dd HH:mm:ss
- **emplacement_pk_start**: Identificador de la estación de inicio (número entero)
- **end_date**: Fecha y hora de la devolución de la bicicleta (fin del viaje) en formato yyyy-MM-dd HH:mm:ss
- **emplacement_pk_end**: Identificador de la estación de fin (número entero)
- **is_member**: Si el usuario del alquiler es miembro del sistema de alquiler (0 si no es miembro, 1 si lo es)

El archivo se compone de una primera línea de encabezado con los títulos de cada campo. De la segunda línea en adelante, cada una representa un viaje de una bicicleta alquilada y devuelta conteniendo los datos de cada uno de los campos, separados por ";".

Ejemplo de las 3 primeras líneas de bikesMON.csv

```
start_date;emplacement_pk_start;end_date;emplacement_pk_end;is_member
2021-09-20 06:31:28;348;2021-09-20 07:02:22;332;1
2022-09-02 11:19:47;753;2022-09-02 11:22:23;702;0
...
```

La segunda línea indica que un usuario **miembro** alquiló una bicicleta **en la estación con id 348** el 20/09/2021 a las 06:31 y la **devolvió en la estación con id 332** el mismo día a las 07:02.

Datos de alquileres de Nueva York , a partir de ahora **bikesNYC.csv**

El archivo CSV tiene la siguiente estructura:

- **started_at**: Fecha y hora del alquiler de la bicicleta (inicio del viaje) en formato yyyy-MM-dd HH:mm:ss
- **start_station_id**: Identificador de la estación de inicio (número entero)
- **ended_at**: Fecha y hora de la devolución de la bicicleta (fin del viaje) en formato yyyy-MM-dd HH:mm:ss
- **end_station_id**: Identificador de la estación de fin (número entero)
- **rideable_type**: Tipo de bicicleta utilizada (cadena de caracteres, "electric_bike", "docked_bike" y "classic_bike")
- **member_casual**: Si el usuario del alquiler es miembro del sistema de alquiler (cadena de caracteres, "casual" si no es miembro, "member" si lo es)

El archivo se compone de una primera línea de encabezado con los títulos de cada campo. De la segunda línea en adelante, cada una representa un viaje de una bicicleta alquilada y devuelta conteniendo los datos de cada uno de los campos, separados por ";".

Ejemplo de las 3 primeras líneas de bikesNYC.csv

```
started_at;start_station_id;ended_at;end_station_id;rideable_type;member_casual
2022-11-17 19:05:10.000000;489509;2022-11-17 19:07:30.000000;490309;classic_bike;member
2022-11-13 13:42:35.000000;702804;2022-11-13 14:24:49.000000;808505;classic_bike;casual
...
```

La segunda línea indica que un usuario **miembro** **alquiló** una bicicleta de tipo **classic_bike** **en la estación con id 489509** el 17/11/2022 a las 19:05 y la **devolvió en la estación con id 490309** el mismo día a las 19:07.

Datos de estaciones de Montreal , a partir de ahora **stationsMON.csv**

El archivo CSV tiene la siguiente estructura:

- **pk**: Identificador de la estación (número entero)
- **name**: Nombre de la estación (cadena de caracteres)
- **latitude**: Latitud de la ubicación de la estación (número real)
- **longitude**: Longitud de la ubicación de la estación (número real)

El archivo se compone de una primera línea de encabezado con los títulos de cada campo. De la segunda línea en adelante, cada una representa una estación, conteniendo los datos de cada uno de los campos, separados por ";".

Ejemplo de las 3 primeras líneas de stationsMON.csv

```
pk;name;latitude;longitude
327;Sanguinet / de Maisonneuve;45.513405;-73.562594
267;Gilford / de Brébeuf;45.530816;-73.581626
...
```

La segunda línea indica que la estación de id **327** se llama **Sanguinet / de Maisonneuve** y se encuentra en la coordenada geográfica (45.513405; -73.562594).

Datos de estaciones de Nueva York , a partir de ahora **stationsNYC.csv**

El archivo CSV tiene la siguiente estructura:

- **station_name**: Nombre de la estación (cadena de caracteres)
- **latitude**: Latitud de la ubicación de la estación (número real)
- **longitude**: Longitud de la ubicación de la estación (número real)
- **id**: Identificador de la estación (número entero)

El archivo se compone de una primera línea de encabezado con los títulos de cada campo. De la segunda línea en adelante, cada una representa una estación, conteniendo los datos de cada uno de los campos, separados por ";".

Ejemplo de las 3 primeras líneas de stationsNYC.csv

```
station_name;latitude;longitude;id
River Ave & E 151 St;40.822217;-73.928939;796704
Madison St & Evergreen Ave;40.69122;-73.91693;456001
...
```

La segunda línea indica que la estación de id **796704** se llama **River Ave & E 151 St** y se encuentra en la coordenada geográfica (40.822217; -73.928939).

Si al procesar un alquiler, éste se refiere a una estación que no está incluida en este archivo (ya sea la estación de alquiler y/o la de devolución), entonces se ignora ese alquiler.


Se asume que todos los alquileres son completos, es decir las bicicletas son alquiladas y devueltas. No es necesario contemplar el caso de bicicletas en circulación, que iniciaron un viaje en una estación pero todavía no fueron devueltas (que no tengan estación de fin ni fecha y hora de devolución)

Se asume que el formato y contenido de los archivos es correcto.

El programa deberá recibir por línea de comando **(no por entrada estándar)** el path del archivo de alquileres y el path del archivo de estaciones.

El ejecutable para procesar los datos de  debe llamarse **bikeSharingMON**.

El ejecutable para procesar los datos de  debe llamarse **bikeSharingNYC**.

Por ejemplo para procesar los datos de , si el archivo CSV de alquileres se llama **bikes.csv**, el de estaciones se llama **stations.csv** y ambos están en el mismo directorio que el ejecutable **bikeSharingMON**, el programa se debe invocar como:

```
$> ./bikeSharingMON bikes.csv stations.csv
```

Si el archivo CSV de alquileres se llama `test.csv` y está en el directorio superior al ejecutable `bikeSharingMON` y el de estaciones se llama `stations.csv` y está en la misma carpeta que el ejecutable se invocará como

```
$> ./bikeSharingMON ../test.csv stations.csv
```

Una vez recibido correctamente el path de los archivos CSV, el programa deberá resolver las *queries* que se listan a continuación, dejando los resultados de cada una en archivos `.csv` y `.html` con el nombre pedido y localizados en el mismo directorio que el ejecutable. Para generar los archivos html deberán usar la librería `CTable` provista por la Cátedra, que pueden descargarla desde [aquí](#).

MUY IMPORTANTE: No se deben hacer cambios a la librería.

Resolver todas las consultas con una única lectura de archivo y de una sola vez. **No se aceptarán implementaciones que ofrezcan un menú de opciones o similar que permita al usuario decidir cuál de las consultas desea procesar.**

Los ejemplos de salida que se listan a continuación tienen valores ficticios y **no necesariamente coinciden con la salida esperada para los archivos CSV.**

En caso de rendir en la primera fecha de final podrán formar grupos de tres o menos alumnos e implementar las tres primeras queries. En caso de formar grupos de cuatro alumnos deberán implementar las cuatro primeras queries.

En caso de rendir en la segunda fecha de final podrán formar grupos de tres o menos alumnos y deberán implementar todas las queries. No se aceptarán grupos de cuatro alumnos en la segunda fecha.

El TPE no debe basarse exclusivamente en los datos de los archivos CSV publicados

Query 1: Total de viajes iniciados por miembros por estación

Donde cada línea del archivo csv de salida contenga, separados por “;” el **nombre de la estación** y la **cantidad total de viajes iniciados en esa estación por usuarios miembros**.

La información debe listarse ordenada en forma descendente por cantidad total de viajes y a igualdad de viajes desempatar alfabéticamente por nombre de la estación.

☐ Nombre del archivo: **query1.csv**

☐ Salida de ejemplo para :

Station;StartedTrips

Métro Mont-Royal (Rivard / du Mont-Royal);89836

Marquette / du Mont-Royal;61421

du Mont-Royal / Clark;59288

Boyer / du Mont-Royal;51431

Métro Atwater (Atwater / Ste-Catherine);51431

...

❏ Salida de ejemplo para 🇺🇸:**Station;StartedTrips**

W 21 St & 6 Ave;22110

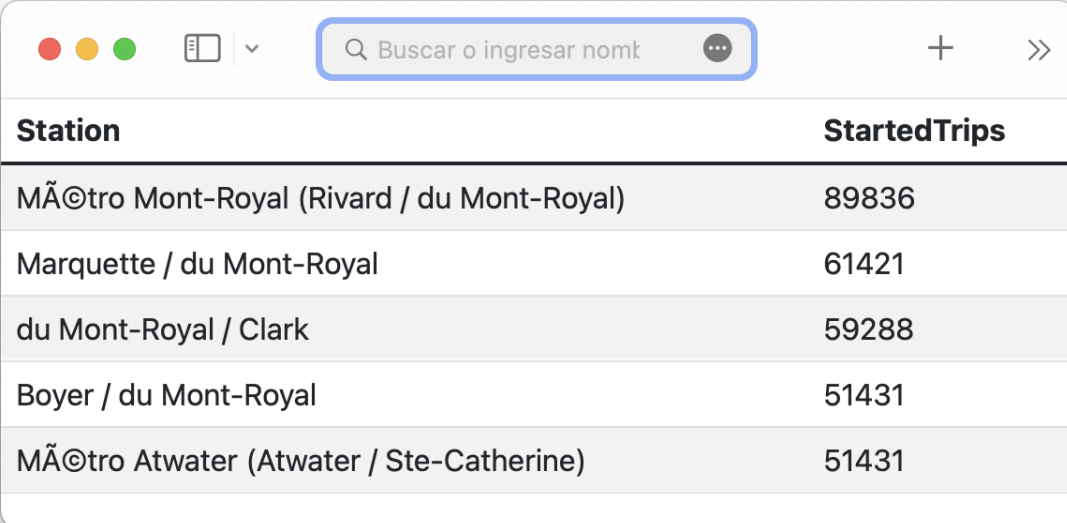
1 Ave & E 68 St;17109

Broadway & W 58 St;16145

University Pl & E 14 St;16145

W 31 St & 7 Ave;14332

...

❏ Nombre del archivo: **query1.html**❏ Salida de ejemplo para 🇨🇦:


The screenshot shows a web browser window with a search bar at the top containing the text "Buscar o ingresar nomk". Below the search bar is a table with two columns: "Station" and "StartedTrips". The table contains five rows of data, which are the first five rows of the output for the Canadian example.


Station	StartedTrips
MÃ©tro Mont-Royal (Rivard / du Mont-Royal)	89836
Marquette / du Mont-Royal	61421
du Mont-Royal / Clark	59288
Boyer / du Mont-Royal	51431
MÃ©tro Atwater (Atwater / Ste-Catherine)	51431

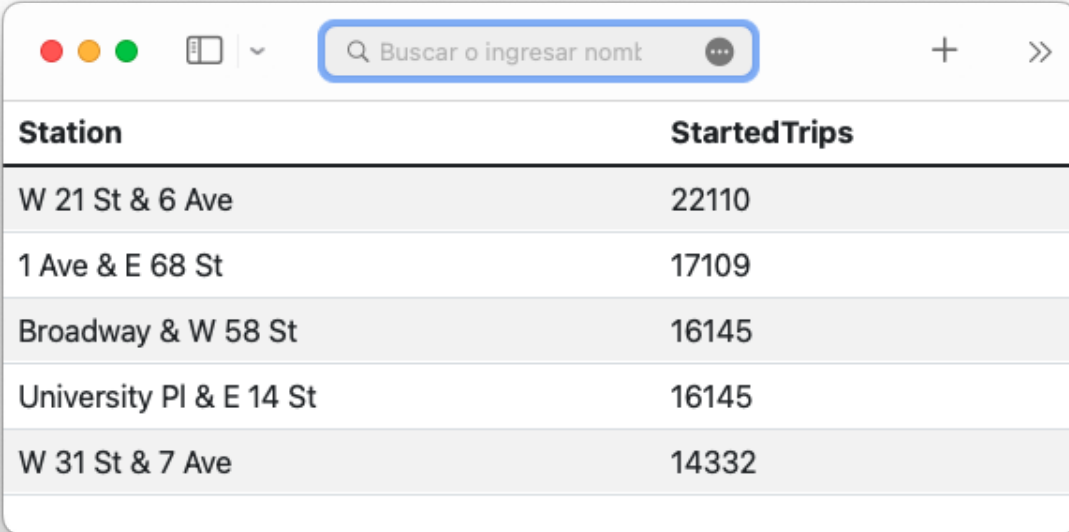
Contenido del archivo query1.html (para este ejemplo se muestran sólo los primeros cinco registros -deben generar todos los registros- y se agregan saltos de línea para facilitar la lectura). No es un problema si algunos caracteres especiales de los nombres de las estaciones no se muestran correctamente (Ver captura anterior).

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-1BmE4kWBq78iYhF1dvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
```



```
crossorigin="anonymous"><html>
  <table class="table table-striped table-hover">
    <thead>
      <tr>
        <th>Station</th>
        <th>StartedTrips</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Métro Mont-Royal (Rivard / du Mont-Royal)</td>
        <td>89836</td>
      </tr>
      <tr>
        <td>Marquette / du Mont-Royal</td>
        <td>61421</td>
      </tr>
      <tr>
        <td>du Mont-Royal / Clark</td>
        <td>59288</td>
      </tr>
      <tr>
        <td>Boyer / du Mont-Royal</td>
        <td>51431</td>
      </tr>
      <tr>
        <td>Métro Atwater (Atwater / Ste-Catherine)</td>
        <td>51431</td>
      </tr>
    </tbody>
  </table>
</html>
```

Salida de ejemplo para :



Station	StartedTrips
W 21 St & 6 Ave	22110
1 Ave & E 68 St	17109
Broadway & W 58 St	16145
University Pl & E 14 St	16145
W 31 St & 7 Ave	14332

Query 2: Total de viajes entre estaciones

Donde cada línea del archivo csv de salida contenga, separados por “;” el **nombre de la estación A**, el **nombre de la estación B**, la **cantidad total de viajes iniciados en la estación A y finalizados en la estación B** y la **cantidad total de viajes iniciados en la estación B y finalizados en la estación A**.

La información debe listarse en orden alfabético por el nombre de la estación A y desempatar alfabéticamente por el nombre de la estación B.

No se deben listar los viajes circulares, es decir donde la estación A y B es la misma.

❑ Nombre del archivo: **query2.csv**

❑ Salida de ejemplo para 🇨🇦:

```
StationA;StationB;Trips A->B;Trips B->A
10e avenue / Holt;10e avenue / Masson;435;535
10e avenue / Holt;12e avenue / St-Zotique;47;83
10e avenue / Holt;15e avenue / Masson;29;18
10e avenue / Holt;16e avenue / Beaubien;29;19
10e avenue / Holt;16e avenue / Jean-Talon;52;61
...
```

❑ Salida de ejemplo para 🇺🇸:

```
StationA;StationB;Trips A->B;Trips B->A
1 Ave & E 110 St;1 Ave & E 18 St;3;4
1 Ave & E 110 St;1 Ave & E 30 St;1;3
1 Ave & E 110 St;1 Ave & E 44 St;2;5
1 Ave & E 110 St;1 Ave & E 62 St;2;2
```



```
1 Ave & E 110 St;1 Ave & E 68 St;3;1
...
```

❑ Nombre del archivo: **query2.html**

Query 3: Total de viajes por mes por estación


Donde cada línea del archivo csv de salida contenga, separados por “;” la **cantidad total de viajes iniciados en el mes de enero** para todos los años, la **cantidad de total de viajes iniciados en el mes de febrero** para todos los años, y **así sucesivamente con todos los meses** y además el **nombre de la estación**.

La información debe listarse en orden alfabético por el nombre de la estación.


En caso de no existir viajes iniciados en algún mes para alguna estación, mostrar un 0.

En caso de no existir viajes iniciados para alguna estación, mostrar un 0 para cada uno de los meses.

❑ Nombre del archivo: **query3.csv**

❑ Salida de ejemplo para :

```
J;F;M;A;M;J;J;A;S;O;N;D;Station
901;905;931;903;904;905;902;901;901;991;911;912;10e avenue / Holt
918;957;967;912;921;963;913;928;989;953;970;992;10e avenue / Masson
956;925;974;910;930;992;944;973;906;989;923;951;11e avenue / du Souvenir
911;946;993;970;902;987;949;917;960;908;997;924;12e avenue / St-Zotique
959;934;983;922;977;911;941;902;915;990;950;921;15e avenue / Masson
...
```

❑ Salida de ejemplo para :

```
J;F;M;A;M;J;J;A;S;O;N;D;Station
901;905;931;903;904;905;902;901;901;991;911;912;1 Ave & E 110 St
918;957;967;912;921;963;913;928;989;953;970;992;1 Ave & E 16 St
956;925;974;910;930;992;944;973;906;989;923;951;1 Ave & E 18 St
911;946;993;970;902;987;949;917;960;908;997;924;1 Ave & E 30 St
959;934;983;922;977;911;941;902;915;990;950;921;1 Ave & E 39 St
...
```

❑ Nombre del archivo: **query3.html**

Para las siguientes *queries* el ejecutable debe recibir adicionalmente hasta dos parámetros más, indicando un rango de años. Las *queries* 4 y 5 deberán considerar únicamente los viajes que hayan sido registrados dentro de ese rango.

En el siguiente ejemplo se consideran todas los viajes

```
$> ./bikeSharingMON bikes.csv stations.csv
```

En el siguiente ejemplo se consideran todos los viajes con una fecha de inicio del año 2000 en adelante

```
$> ./bikeSharingMON bikes.csv stations.csv 2000
```

En el siguiente ejemplo se consideran todas los viajes con una fecha de inicio del año 2000 y hasta el año 2010 inclusive

```
$> ./bikeSharingMON bikes.csv stations.csv 2000 2010
```

En el siguiente ejemplo se consideran sólo los viajes con una fecha de inicio del año 2000

```
$> ./bikeSharingMON bikes.csv stations.csv 2000 2000
```

En el siguiente ejemplo se consideran todos los viajes entre 2023 y 2030 inclusive, por lo que seguramente no haya viajes en esos años y las queries se generen vacías (sólo conteniendo los títulos)

```
$> ./bikeSharingMON bikes.csv sensors.csv 2023 2030
```

En los siguientes ejemplos se informa en salida de errores que los parámetros son incorrectos y no se procesa la información.

```
$> ./bikeSharingMON bikes.csv stations.csv 2000 1990
```

```
$> ./bikeSharingMON bikes.csv stations.csv dosmil 2010
```

```
$> ./bikeSharingMON 2000 2010 bikes.csv stations.csv
```

Query 4: Total de viajes circulares por estación

Donde cada línea del archivo csv de salida contenga, separados por “;” el **nombre de la estación** y la **cantidad total de viajes circulares de esa estación**.

Se considera que un viaje es circular cuando la estación de inicio y la estación de finalización es la misma.

La información debe listarse ordenada en forma descendente por cantidad total de viajes y a igualdad de viajes desempatar alfabéticamente por nombre de la estación.

❑ Nombre del archivo: **query4.csv**

❑ Salida de ejemplo para :

Station;RoundingTrips

Parc Jean-Drapeau (Chemin Macdonald);16321


Crescent / de Maisonneuve;8341

de la Commune / Place Jacques-Cartier;8341

de la Commune / St-Sulpice;7500

Métro Mont-Royal (Rivard / du Mont-Royal);4599

...

- ☐ Salida de ejemplo para :

Station;RoundingTrips
 Grand Army Plaza & Central Park S;2066
 Broadway & W 58 St;1524
 Central Park S & 6 Ave;1524
 7 Ave & Central Park South;1363
 West St & Chambers St;1188
 ...

- ☐ Nombre del archivo: **query4.html**

Query 5: Cantidad de días de afluencia neta positiva, neutra y negativa por estación

Donde cada línea del archivo csv de salida contenga, separados por “;” el **nombre de la estación**, la **cantidad de días con afluencia neta positiva de esa estación**, la **cantidad de días con afluencia neta neutra de esa estación** y la **cantidad de días con afluencia neta negativa de esa estación** para los días pertenecientes al rango de años indicado más arriba.

Por ejemplo, si el rango es 2008 2010, se deben considerar los días entre el 01/01/2008 y el 31/12/2010. Si el rango es 2000, se deben considerar los días entre el 01/01/2000 y el 31 de diciembre del año de la medición más reciente procesada. Si no se ingresa ningún rango se deben considerar los días entre el 1 de Enero del año de la medición más antigua procesada y el 31 de diciembre del año de la medición más reciente procesada.

La afluencia neta de una estación consiste en el total de viajes que finalizan en la estación menos el total de viajes que inician en la estación.

El orden de impresión es alfabético por nombre de la estación.

La suma de los tres valores de afluencia neta debe coincidir con el total de días comprendidos en el rango de fechas que se indica por parámetro. Si para un día en particular no se iniciaron viajes en una estación y no finalizaron viajes en esa estación entonces el día se contabiliza como afluencia neta neutra.

- ☐ Nombre del archivo: **query5.csv**

- ☐ Salida de ejemplo para :

Station;PosAfflux;NeutralAfflux;NegativeAfflux
 10e avenue / Holt;15;3;13
 10e avenue / Masson;13;9;9
 11e avenue / du Souvenir;13;15;3
 12e avenue / St-Zotique;11;4;16
 15e avenue / Masson;7;20;4
 ...

- ☐ Salida de ejemplo para :

Station;PosAfflux;NeutralAfflux;NegativeAfflux

```
1 Ave & E 110 St;15;3;13
1 Ave & E 16 St;13;9;9
1 Ave & E 18 St;13;15;3
1 Ave & E 30 St;11;4;16
1 Ave & E 39 St;7;20;4
...
```

❑ Nombre del archivo: **query5.html**

Para todas las queries, tener en cuenta que todos los archivos de salida deben contener la línea de encabezado correspondiente tal como se indica en las salidas de ejemplo, respetando nombre y orden.

Las salidas de ejemplo de cada query presentadas en esta consigna fueron obtenidas con subconjuntos del dataset provisto por lo que no deben comparar el resultado obtenido con los valores expresados en las salidas de ejemplo. Estos resultados son sólo a modo explicativo para que queden ejemplificados los criterios de ordenación y formato esperado de los resultados.

3. Contenidos

Para la realización del presente trabajo se recomienda consultar el siguiente material:

- Para leer argumentos por línea de comandos:
https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- `strtok` puede resultar útil para "parsear" las líneas de los archivos:
https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
- **Presentación de Archivos:** Contiene lo esencial para el manejo de archivos en C. Tener en cuenta que se procesarán únicamente archivos de texto en forma secuencial, por lo que las funciones a utilizar serán `fopen`, `fgets`, `fclose`, `fprintf` o similares. Se encuentra en Campus ITBA.

De todas formas, los alumnos pueden consultar dudas relacionadas a estos nuevos contenidos. Ver sección 9. *Dudas sobre el TPE.*

4. Diseño e Implementación del Programa

Se debe realizar un diseño donde se separe claramente la lectura y preparación de datos (*front-end*) del almacenamiento y procesamiento de los datos (*back-end*).

Recordar que ninguna función de *back-end* debe invocar a una función de *front-end*.

Diseñar e implementar el backend pensando que alguien más podrá hacer otro front-end.

Para el almacenamiento de los datos en memoria se deberá desarrollar al menos un TAD.

En ningún caso se debe repetir código para resolver situaciones similares, sino que debe implementarse una correcta modularización y se deben reutilizar funciones parametrizadas.

El *back-end* debería servir también para otros *front-ends*, como por ejemplo mostrar la información por pantalla, en forma de gráficos, o que la información sea leída por teclado o desde una base de datos.

Tanto la biblioteca como el *front-end* deben estar correctamente comentados y en el caso de la biblioteca se debe escribir el archivo de encabezado correspondiente.

El programa no debe presentar leaks de memoria.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado. En todo caso será el *front-end* (nunca el *back-end*) el que -ante un error- podrá decidir que no tiene sentido seguir ejecutando la aplicación.

Se tendrá en cuenta la eficiencia en el uso de recursos, tanto el tiempo de ejecución como la memoria utilizada.

A diferencia de lo pedido en los parciales, se debe validar que todas las funciones de la familia *alloc* (*malloc*, *calloc*, *realloc*) hayan podido reservar la memoria necesaria.

5. Uso de Git

Es obligatorio el uso de un repositorio Git para la resolución de este final. Siguiendo los pasos indicados en la clase de Taller de Git deberán crear un repositorio en GitHub donde todos los integrantes del grupo colaboren con las modificaciones del código provisto. **No se aceptarán entregas que utilicen un repositorio *git* con un único *commit* que consista en la totalidad del código a entregar.**

Los distintos *commits* deben permitir ver la evolución del trabajo, tanto grupal como **individual**

Muy importante: los repositorios creados deben ser privados.

6. Material a entregar

Cada grupo deberá subir al Campus ITBA un archivo compactado conteniendo como mínimo los siguientes archivos:

- Archivos fuentes y de encabezado.
- Archivo de texto README con la explicación de cómo generar los ejecutables y de cómo ejecutarlos.
- *makefile*. Deberá generar el ejecutable pedido. Consultar apunte en Campus ITBA.

Revisar que en el archivo comprimido también se encuentra el directorio oculto *.git/* donde se detallan todas las modificaciones realizadas.

7. Armado de Grupos

Los alumnos deben informar la conformación del grupo y la fecha de final elegida en el Debate “TPE Final: Conformación Grupos” de Campus ITBA **al menos un día antes de la fecha de entrega correspondiente**.

Cada grupo deberá realizar la entrega mediante la actividad correspondiente en Campus ITBA:

- En caso de rendir en la primera fecha de final (14/07/2023) deberán entregar antes del 11/07/2023 23:59 ART.
- En caso de rendir en la segunda fecha de final (21/07/2023), deberán realizar la entrega antes del 18/07/2023 23:59 ART.

En todos los casos deben enviar sólo el material pedido. No incluir código compilado, archivos de prueba publicados, etc.

El grupo completo deberá presentarse en un día y hora previamente acordada (se asignarán turnos para todos los grupos que rindan el final ese día). En ese encuentro se comunicará la nota del final y los integrantes del grupo rendirán un coloquio, el cual podrá modificar la nota individual de los integrantes del grupo.

Para que el trabajo sea aceptado todos los integrantes del grupo deben estar inscriptos en fecha de final que acordaron en el armado del grupo.

En caso de inscribirse y no entregar en fecha, se calificará como ausente. Si algún miembro del equipo no se presenta en el horario previamente acordado el alumno será calificado como ausente, y deberá conformar otro grupo en otra fecha de final, no afectando la evaluación de los alumnos que se presenten.

Todos los integrantes de cada grupo deberán inscribirse por su cuenta en la fecha de final elegida en el SGA.

8. Criterios de Evaluación y Calificación

El programa debe poder compilarse y ejecutarse en Pampero usando **gcc** con los parámetros **-pedantic -std=c99 -Wall -fsanitize=address**.

Recomendamos utilizar todos los flags desde el principio del desarrollo para encontrar y resolver antes los posibles errores que surjan. Es útil agregar además el flag de *debug* -g para obtener más información sobre los errores encontrados por el -fsanitize.

No se aceptará el uso de bibliotecas de terceros, a excepción de la biblioteca estándar de C. El código debe ser íntegramente de autoría propia. El uso de bibliotecas no autorizadas implicará la desaprobación.

Si un trabajo presenta errores de compilación, el mismo será reprobado. Se espera que no se presenten “warnings” evitables.

Para la evaluación y calificación del trabajo especial se considerarán:

- el correcto diseño de el/los TAD, que debería poder ser fácilmente adaptable para otras codificaciones, nombres o formatos de archivos
- la separación del *front-end* y el *back-end* (recordar que el back-end no debe saber cuál es origen o el destino de los datos, como tampoco el formato en el cual se reciben los datos o escriben los resultados)
- el cumplimiento del modo de ejecución y de los nombres de archivos pedidos
- el correcto funcionamiento del programa

- la modularización realizada
- la claridad del código
- el cumplimiento de las reglas de estilo de programación dadas en clase
- la eficiencia del procesamiento de los datos y el uso de la memoria.

9. Dudas sobre el TPE

Si bien el enunciado contempla la funcionalidad completa a desarrollar es normal que surjan dudas acerca de cómo interpretar ciertos casos. O que una consigna genere más de una posible solución, por lo que es importante que analicen bien el enunciado, y ante cualquier duda pregunten. Sólo se contestarán dudas sobre el enunciado. Las mismas deben volcarse en Github: <https://github.com/PI-ITBA/TPE/issues>

En caso de realizar alguna aclaración o consideración sobre el enunciado, la misma deberá ser tomada en cuenta por todos los grupos, no solo para el grupo que haya hecho la pregunta.