

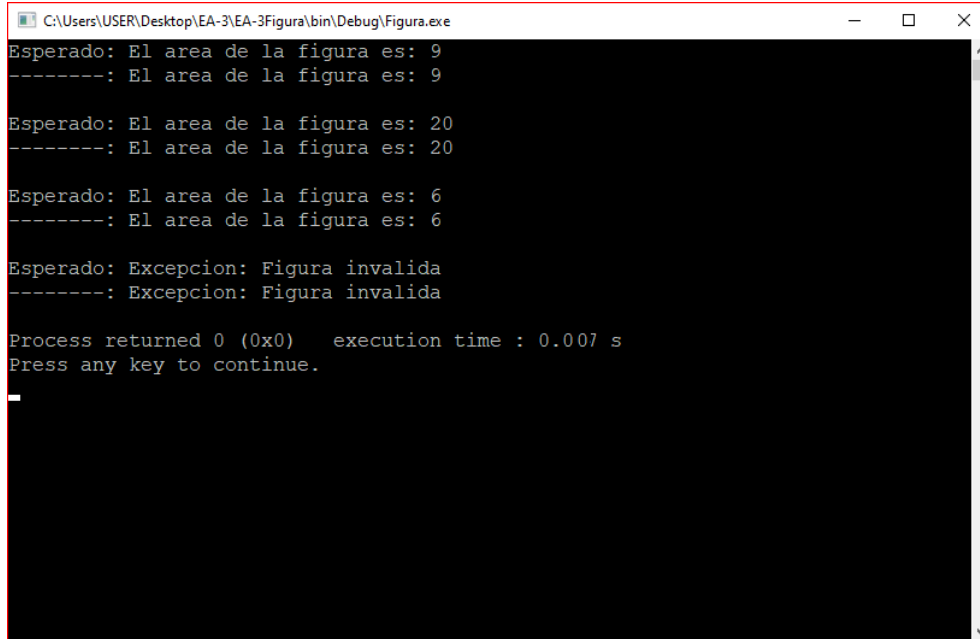
**E. A. Nro. 3– Tercera Evaluación de Aprendizaje.**

Parte 1,-.POO.- Un muy simple y muy conceptual ejercicio de Programación Orientada a Objetos en Lenguaje C++, en el que demuestre los conocimientos adquiridos de herencia, excepciones, etc.

El proyecto que se le entrega contiene solo el archivo main.cpp y debe agregar los archivos con la declaración de las clases y sus desarrollos.

```
main.cpp X
10 void funcionMostrar(Figura *p);
11
12 int main()
13 {
14     cout << "Esperado: El area de la figura es: 9" << endl << "-----: ";
15     Cuadrado *c = new Cuadrado(3);
16     funcionMostrar(c);
17
18     cout << endl << "Esperado: El area de la figura es: 20" << endl << "-----: ";
19     Figura *r = new Rectangulo(4, 5);
20     funcionMostrar(r);
21
22     cout << endl << "Esperado: El area de la figura es: 6" << endl << "-----: ";
23     Triangulo t(3, 4);
24     funcionMostrar(&t);
25
26     // La siguiente linea no deberia compilar
27     //Figura f();
28
29     cout << endl << "Esperado: Excepcion: Figura invalida" << endl << "-----: ";
30     try
31     {
32         Cuadrado malC(0);
33         funcionMostrar(&malC);
34         Triangulo *malT = new Triangulo (-1, -5);
35         funcionMostrar(malT);
36     }
37     catch(FiguraInvalidaException &efi)
38     {
39         cout << "Excepcion: " << efi.what() << endl;
40     }
41
42     delete c;
43     delete r;
44
45     return 0;
46 }
47
48 void funcionMostrar(Figura *p)
49 {
50     cout << "El area de la figura es: " << p->area() << endl;
51 }
52
```

La salida por pantalla correspondiente es:



```
C:\Users\USER\Desktop\EA-3\EA-3Figura\bin\Debug\Figura.exe
Esperado: El area de la figura es: 9
-----: El area de la figura es: 9

Esperado: El area de la figura es: 20
-----: El area de la figura es: 20

Esperado: El area de la figura es: 6
-----: El area de la figura es: 6

Esperado: Excepcion: Figura invalida
-----: Excepcion: Figura invalida

Process returned 0 (0x0)   execution time : 0.007 s
Press any key to continue.
```

ATENCIÓN: en la E. A. R. Nro. 3 se evaluarán además algunos temas de Java-

Parte 2.-TDA.- Se dispone de una función que permite cargar información sintética (simulada) de los exámenes finales de alumnos en un TDA LISTA implementado en una lista dinámica doblemente enlazada. Esta información está compuesta de: legajo (el DNI), apellido(s) y nombre(s), código de la materia y calificación obtenida, almacenados en orden cronológico en la lista.

Se requiere:

- La primitiva que muestra la lista del primero al último, similar a las dadas, mostrando los títulos (sólo si hay datos a mostrar).
- La primitiva que ordene la lista según la frecuencia de aparición de los nodos (al comienzo quedarán aquellos cuya clave está más veces pero respetando el orden cronológico -para más detalles, ver al final-).



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

- La primitiva que elimina todos los nodos, generando un listado en que para cada alumno se muestre al final la cantidad de materias rendidas y su promedio en el mismo orden en que estaban cargados en la lista y que devuelva la cantidad de nodos eliminados.

Para que las primitivas anteriores puedan cumplir su tarea, se deben resolver las funciones de manejo de la información (o datos) cargados en la lista.

Para el ordenamiento pedido, bajo ningún concepto debe utilizar otras primitivas. Pero sí puede utilizar, si lo necesita, otras funciones auxiliares. No debe crear ni eliminar nodos.

La salida de su programa debe coincidir exactamente con la que hace el proyecto que se le entrega. No debe alterar los archivos `funciones.h` ni `main.h`. Las únicas modificaciones en `main.c` son para que invoque sus primitivas. Cualquier `"include"` de biblioteca estándar que necesite, debe hacerlo en `funciones.c`.

Dispone de un proyecto que al ejecutarlo hace todo lo pedido, con lo que puede comparar los resultados que obtiene con los resultados esperados.

Consideraciones Generales (idénticas a las evaluaciones anteriores)

Descargue el proyecto provisto, genere una *carpeta* en su computadora y en ella descomprímalo.

Verá que se generan dos *subcarpetas*, una para plataforma (o compilador) de 32 bits y otro para plataforma de 64 bits.

En la *carpeta* que corresponde a su IDE (**EA3-Proyecto64** o **EA3-Proyecto32**) abra el proyecto que corresponde a su plataforma (habitualmente con doble click en el archivo **EA-3.cbp** o en su defecto una vez abierto el IDE, "*arrastrándolo*" en el mismo).

Proceda a compilarlo.



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

Si en el cuadro de diálogo inferior le aparecen errores:

... ¡¡¡es porque se equivocó de proyecto!!! Vaya al otro y compile:

Cuando compile el proyecto, asegúrese que el IDE genera el ejecutable con las opciones como se ve en la figura anterior.

```
gcc.exe -Wall -g -c C:(...etcétera)
gcc.exe -Wall -g -c C:(...etcétera)
gcc.exe -o bin\Debug\(...etcétera) .\lib-EA_3_Release_bib.a
```

De otro modo, podrá tener algunos problemas en los que no le podremos ayudar.

DEBE DEVOLVER TAN SOLO el archivo "**funciones.c**", además de los archivos "**.cpp**" y "**.h**" correspondientes al ejercicio de POO-C++ y al comienzo de cada uno de estos archivos, debe poner en una línea de comentario:

DNI-APELLIDO,Nombre-(curso-comisión), por ejemplo:

```
/**/* 22.333.444-PEREZ_DEL_RIO, JuanManuel- (07-2299) */**/
```

Estos archivos deben estar comprimidos únicamente en formato **.zip (*)** . Para ello copie los tres archivos en una misma ubicación, selecciónelos y con botón derecho elija de la ventana emergente la opción: **[Enviar a] / [Carpeta comprimida (en zip)]**
(o **[Send to] / [Compressed (zipped) folder]**)

El archivo comprimido debe estar renombrado con su número de dni sin puntos, por ejemplo: **22333444.zip**

Aclaración sobre qué significa y qué se espera al ordenar de mayor a menor un TDA-LISTA por la frecuencia de aparición de la clave, respetando el orden cronológico con que fue cargado en la lista.

Este ordenamiento no es un caso trivial, por esto vale que lo orientemos en la primera etapa: entender el problema.

Graficaremos una de las posibles estrategias para el ordenamiento pedido tomando como ejemplo lo siguiente: La información (o datos) a ordenar tiene una clave (la letra) además de otros miembros de esa información (o de ese dato), que graficaremos con el número.



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

Partiendo por ejemplo de . . .

A-6 B-2 K-2 D-8 A-9 F-2 C-8 F-1 N-8 K-5 F-6 K-3 A-3 D-5 N-5 A-5 K-7 D-5 F-4 B-7

Se busca el primero que aparezca más veces . . .

A-6 B-2 K-2 D-8 A-9 F-2 C-8 F-1 N-8 K-5 F-6 K-3 A-3 D-5 N-5 A-5 K-7 D-5 F-4 B-7



. . . que en este caso es la clave "A" que está 4 veces, con lo que los cuatro se pasarán al comienzo (respetando el orden relativo en que estaban) quedando . . .

B-2 K-2 D-8

F-2 C-8 F-1 N-8 K-5 F-6 K-3

D-5 N-5

K-7 D-5 F-4 B-7

A-6 A-9 A-3 A-5

Se vuelve a buscar el que más veces aparece desde el principio . . .

B-2 K-2 D-8

F-2 C-8 F-1 N-8 K-5 F-6 K-3

D-5 N-5

K-7 D-5 F-4 B-7



A-6 A-9 A-3 A-5

. . . resultando el que tiene la clave "K" con cuatro ocurrencias, con lo que pasarán a estar a continuación de los anteriores . . .

B-2

D-8

F-2 C-8 F-1 N-8

F-6

D-5 N-5

D-5 F-4 B-7

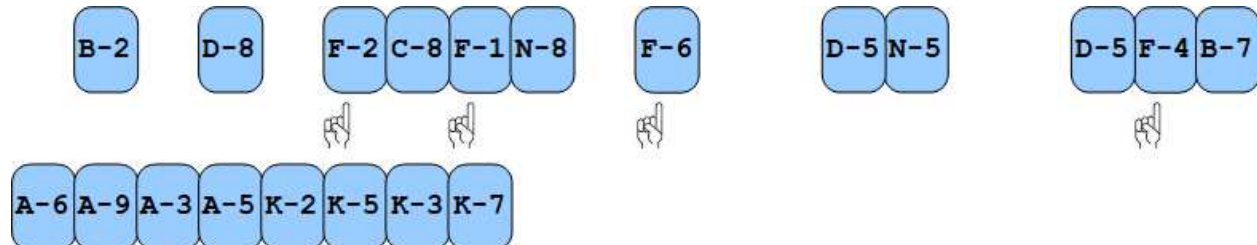
A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7

Se vuelve a buscar el que más veces aparece desde el principio . . .

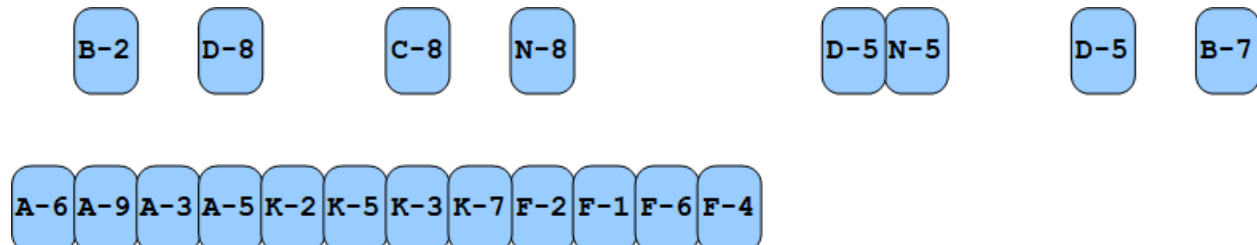


UNLaM

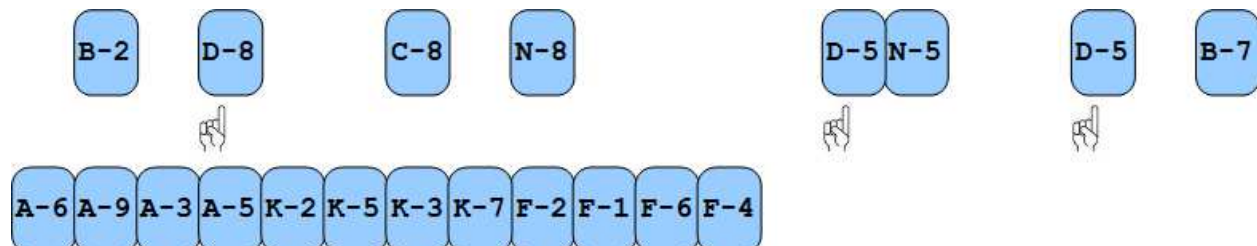
Dto. Ingeniería e Investigaciones Tecnológicas



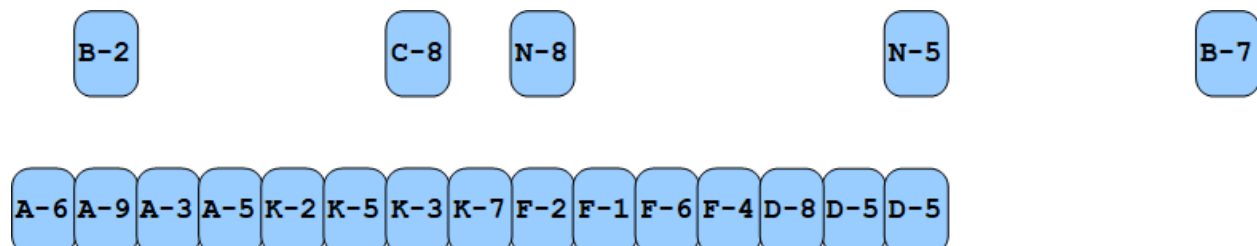
... resultando el que tiene la clave "F" con cuatro ocurrencias, con lo que pasarán a estar a continuación de los anteriores ...



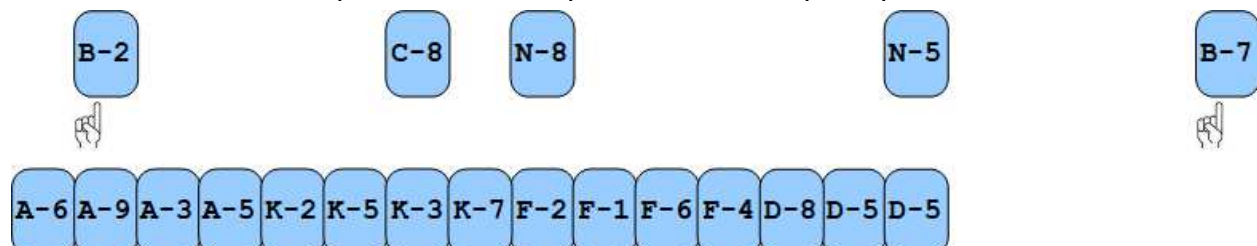
Se vuelve a buscar el que más veces aparece desde el principio ...



... resultando el que tiene la clave "D" con tres ocurrencias, con lo que pasarán a estar a continuación de los anteriores ...



Se vuelve a buscar el que más veces aparece desde el principio ...





UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

... resultando el que tiene la clave "B" con dos ocurrencias, con lo que pasarán a estar a continuación de los anteriores ...

C-8

N-8

N-5

A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7 F-2 F-1 F-6 F-4 D-8 D-5 D-5 B-2 B-7

Se vuelve a buscar el que más veces aparece desde el principio ...

C-8

N-8

N-5

A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7 F-2 F-1 F-6 F-4 D-8 D-5 D-5 B-2 B-7

... resultando el que tiene la clave "N" con dos ocurrencias, con lo que pasarán a estar a continuación de los anteriores ...

C-8

A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7 F-2 F-1 F-6 F-4 D-8 D-5 D-5 B-2 B-7 N-8 N-5

Se vuelve a buscar el que más veces aparece desde el principio ...

C-8

A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7 F-2 F-1 F-6 F-4 D-8 D-5 D-5 B-2 B-7 N-8 N-5

... resultando el que tiene la clave "C" con una ocurrencia, con lo que pasará a estar a continuación de los anteriores ...

A-6 A-9 A-3 A-5 K-2 K-5 K-3 K-7 F-2 F-1 F-6 F-4 D-8 D-5 D-5 B-2 B-7 N-8 N-5 C-8



UNLaM

Dto. Ingeniería e Investigaciones Tecnológicas

. . . y ya no quedan más, quedando de este modo el conjunto ordenado.

ATENCIÓN: en ningún momento, al ordenar la lista, debe modificar la variable lista (cual es en nodo que el TDA-LISTA direcciona -es una lista dinámica doblemente enlazada- y no necesariamente será ni al primero ni al último), si originalmente apuntaba al último insertado (en nuestro ejemplo "B-7") debe quedar apuntando al nodo al que apuntaba sin modificación.

Lo hemos ayudado a entender el problema y además hemos esbozado una de las estrategias posibles con que lo debe resolver.

Deberá resolverlo respondiendo a esta estrategia.

No debe eliminar ni crear nodos, tan sólo debe reordenarlos.

En la función "**main**" sólo debe invocar a sus funciones en los tres lugares indicados.

Ya sabe que el proyecto se entrega funcionando con las primitivas y funciones solicitadas totalmente operativas.

Su salida por pantalla (o en el archivo que se genera) deberá coincidir exactamente con la salida que produce el programa que le entregamos.

Su solución debe compilar sin advertencias ("**warnings**").