UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

Rehacer Evaluación de Aprendizaje (REdA3).

Lea atentamente este documento antes de comenzar a resolver la ejercitación planteada.

En este Rehacer de la 3er. EdA, la cátedra ha coincidido que los alumnos que no la han aprobado la deberán resolver en su totalidad.

Esta será la condición mínima para quedar regular en la materia.

Para promocionar deberá resolver un nuevo ejercicio de TDA, un muy simple y elemental ejercicio de POO/C++, además de un ejercicio de Java.

Debe resolver la siguiente problemática eligiendo la mejor estrategia.

TDA – Lista doblemente enlazada – Generar el "*podio*" con la nMejoresCalificaciones.
Acerca de qué se trata el problema.

En una organización le encargan a un programador una función que permita determinar las <**nMejores**> calificaciones en una evaluación. Se le explica que esa evaluación puede tener puntajes o calificaciones (<**calif**>) del 0 al 10 ó cualquier otra escala de puntajes, que no serían precisamente calificaciones (podría ser el resultado de una evaluación de opciones múltiples o "*múltiple choice*" tomado mediante un "*form*" de "*Teams*" con puntajes, por ejemplo, de 0 a 40).

Especificación funcional:

Los datos a procesar, que se irán ingresando de a uno por vez, responden a un tipo de dato que contiene una clave (<**clave**> alfanumérica de 7 caracteres válidos que identifican al evaluado) y un puntaje (<**calif**> entero que refleja la calificación o puntaje obtenido). Se requiere una muy buena velocidad de procesamiento y de aprovechamiento de memoria, por lo que se indica el uso de un TDA lista en que se eliminen los elementos cargados con anterioridad que ya no deben estar en la misma para luego almacenar el nuevo elemento de modo que queden ordenados por puntaje (de mayor a menor) y a igualdad de puntaje ordenados por la clave de menor a mayor (comparación lexicográfica).

Ejemplo: suponga que ya se han cargado algunos y que se buscan los tres mayores puntajes:

J-4 / P-4 / F-3 / J-2

Si ahora vinieran: **S-1** ó **T-0** no se pondrían en la lista porque ya están los tres mayores puntajes (hasta el momento).

Si en cambio se trataba de: **H-2** compartiría el 3er lugar con **J-2**

J-4 / P-4 / F-3 / H-2 / J-2

Si luego vinieran más datos con 2 compartirían el 3er lugar:

J-4 / P-4 / F-3 / A-2 / H-2 / I-2 / J-2 / M-2

Si vinieran más con cualquier puntaje de **4** a **2** quedarían compartiendo lugar con los preexistentes.

J-4 / M-4 / P-4 / Q-4 / B-3 / F-3 / S-3 / A-2 / D-2 / H-2 / I-2 / J-2 / M-2

Si ahora viniera uno con mayor calificación, por ejemplo, **R-6**, que es superior a las que había

- en 1er lugar se deben eliminar los que ya no estarán quedando:

J-4 / M-4 / P-4 / Q-4 / B-3 / F-3 / S-3

- y luego se insertará la nueva evaluación

R-6 / J-4 / M-4 / P-4 / Q-4 / B-3 / F-3 / S-3

De acá en adelante no se pondrán en la lista las calificaciones inferiores a **3** y se agregarán las que vengan con **6**, **4** y **3**, hasta que llegue una calificación superior a **6** ó igual a **5** con lo que se eliminarán todos los que tengan **3** para insertar el nuevo.

Tipos de Datos Abstractos

Punto 1.-

Escriba una primitiva <**ponerEnListaDoblNCalif**> del TDA-Lista implementada con memoria dinámica en una lista doblemente enlazada que resuelva el problema planteado. Las comparaciones necesarias para cumplir su propósito deberán hacerse invocando a la(s) función(es) que resuelvan la comparación. En este caso, no las recibe por argumento. Eso se hará en una segunda etapa por parte del equipo de la empresa que contrata su trabajo.

Las comparaciones deben respetar el criterio de ordenamiento, devolviendo 0 (cero) si lo que se compara coincide. Algún valor negativo si al primero que interviene en la comparación le corresponde estar antes que al segundo. De lo contrario un valor positivo.

Ponga especial atención a lo pedido, es simple pero no es trivial.

Para la lista doblemente enlazada, tal como se vio en clase y apuntes, debe emplear puntero a nodo y en caso de insertar un nuevo nodo la variable "*lista*" de "*main*" debe quedar con la dirección del último insertado.

Debe utilizar puntero a nodo para recorrer la lista simplemente enlazada, tal como se vio en clase y apuntes de la materia.

La función devolverá la cantidad de nodos eliminados (con un valor entero negativo), si no hubiera eliminación utilice la etiqueta **`TODO_BIEN`**, en caso de falta de memoria utilice **`SIN_MEMORIA`**. Se garantiza que no se dará clave duplicada por lo que no hay que tenerla en cuenta. Si no se puede insertar (porque ya hay cuatro -o n- mejores), deberá devolver con la etiqueta **`NO_SE_INSERTA`**.

NO se aceptará que recorra dos veces la lista, ni que inserte antes de eliminar nodos sobrantes (si los hubiera). La lista NO se debe ordenar, se genera ordenada tal como se indicó.

NOTA: tenga en cuenta que es válido ir al primero de la lista sin contar ni contabilizar.

Debería considerar generar su propio lote de pruebas además del provisto con el proyecto asegurándose la correcta ejecución de lo pedido.

POO/C++ – Operar con Racionales (prueba para ingresar a un trabajo)
Acerca de qué se trata el problema.

En una selección de candidatos a ocupar un puesto de programador en una empresa, le entregan un programa en que se realizan unas pocas operaciones con objetos que modelan números racionales.

El programa en cuestión cumple con las pocas operaciones pedidas, para el lote de prueba que puso el aspirante, pero adolece de serias fallas conceptuales y de cálculo.

Lo más grave, es que no está contemplado el caso de denominador cero.

Debe resolver las correcciones necesarias para un correcto diseño de la clase y que permita funcionar con cualquier par de numerador y denominador.

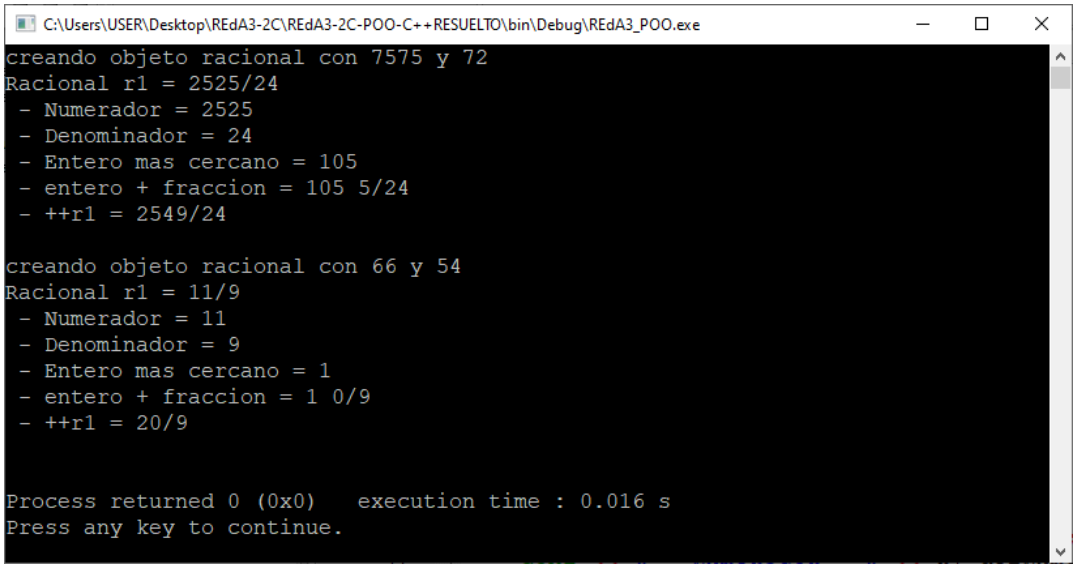
POO-C++

Punto 2.-

Se le entrega un proyecto completo compuesto de '**`main.cpp`**', '**`main.h`**', '**`clase.h`**' y '**`clase.cpp`**'. El mismo compila y ejecuta correctamente con las pruebas hechas en la función '**`main`**', adoleciendo de errores que deben ser corregidos.

Debe corregir lo hecho por la misma ya que tiene "*problemas*" con otros modos de uso (numeradores y denominadores).

Salida por pantalla del Proyecto de POO/C++



POO/Java – Operaciones con la clase "Automovil"
POO-Java

Crear una clase "**`Automovil`**" con los siguientes atributos:

- **`marca`**

- **`modelo`**

- **`kilometraje`**

Además, esta clase debe tener un atributo estático denominado **`kilometrajeTotal`**, en el que se acumulará el total de kilómetros recorridos por todos los automóviles creados.

Resolver los constructores y métodos MINIMOS Y NECESARIOS para hacer funcionar el siguiente main:

```
    public static void main(String[] args) {
//        - Solicitar por teclado el año del auto de Luis
//        - Solicitar por teclado marca del auto de Luis
//        - Solicitar por teclado modelo del auto de Luis
//        - Solicitar por teclado el año del auto de Juan
//        - Solicitar por teclado marca del auto de Juan
//        - Solicitar por teclado modelo del auto de Juan
//
        Automovil autoDeLuis = new Automovil("Chevrolet", "Cruze",2020);
        Automovil autoDeJuan = new Automovil("Toyota", "Yaris",2017);

        autoDeLuis.recorre(30);
        autoDeLuis.recorre(40);
        autoDeLuis.recorre(220);
        autoDeJuan.recorre(60);
        autoDeJuan.recorre(150);
        autoDeJuan.recorre(90);
        System.out.println(autoDeJuan.toString());
        System.out.println(autoDeLuis.toString());
        System.out.println("El coche de Luis ha recorrido " + autoDeLuis.getKilometraje() + "Km");
        System.out.println("El coche de Juan ha recorrido " + autoDeJuan.getKilometraje() + "Km");
        System.out.println("El kilometraje total ha sido de " + Automovil.getKilometrajeTotal() + "Km");

    }
```

Aclaración: donde dice: "**`solicitar por teclado ...`**" se debe hacer lo necesario para pedir por teclado al usuario los datos que se solicitan en el orden correspondiente (año, marca y modelo).

UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

¿Qué y cómo debe entregar?

En un único archivo (o carpeta comprimida) 22.333.444.zip (reemplazando 22.333.444 por su número de documento) creará tres carpetas:

- EdA3 – en esta carpeta copiará **`funciones.c`**, **`que_hice.h`** además de los directorios **`include`** y **`source`** correspondientes a la EdA3.

- REdA3 – en esta carpeta copiará **`funciones.c`**, **`que_hice.h`** además de los directorios **`include`** y **`source`** correspondientes a la REdA3.

- Java – en esta carpeta copiará la solución al ejercicio pedido.

Recuerde que sólo se aceptan formatos de compresión .zip y lo puede crear con botón derecho mediante: **`[new] / [compressed (zipped) folder] ó`**

`[nuevo] / [carpeta comprimida (zipeada)]`.

Plazo de entrega: viernes 11/12/2020 – 23:59:59 horas.