UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

(0612) PROGRAMACIÓN II
(1110) PROGRAMACIÓN
FINAL DICIEMBRE
MESA A
14 hs
20/12/ 2021

Apellido y Nombre:

DNI: PARCIAL:

Calificación :

Ejercicio C

El departamento de compras de la empresa “Mi Empresa”, cada mes procesa un archivo de texto “pedidos.txt” donde cada línea representa un pedido de un producto. Es un archivo de texto de longitud variable con ‘|’ como separador de campos:

```
“nroPedido”: número de pedido (entero);

“codPro”: código del producto (alfanumérico de 10 caracteres como máximo”;

“cantPedida”: cantidad de unidades del producto solicitadas (entero);

“codSector”: código del sector que realizó el pedido (1 carácter);

“oficina”: oficina que realiza el pedido (alfanumérico de 20 caracteres);
```

Se pide:

- crear dos listas
  - una lista de **compras:** ordenada por código de producto, que acumula la cantidad solicitada total.
  - otra lista de **sectores**: ordenada por sector y producto que acumula la cantidad de cada producto solicitada por cada sector.
- Se le solicita que implemente la primitiva eliminarUltimosNdeLaLista. La función eliminará la cantidad N de nodos del final de la lista. Los parámetros que la función recibe son: lista, un entero N que indica la cantidad de nodos a eliminar. La función retorna 1 (uno) si la operación fue exitosa y 0 (cero) en caso contrario. Tenga en cuenta que si N es mayor a la cantidad de nodos contenidos en la lista la operación no se puede realizar. Para realizar la operación, la lista solo puede ser recorrida completamente UNA SOLA VEZ.
  Tenga en cuenta que esta primitiva se puede utilizar en cualquier aplicación. No presuponga tener ningún conocimiento sobre la lista.
- Genere un archivo binario denominado “resultadoCompras.bin” con el contenido de la lista luego de ejecutar la función del punto b sobre la lista del punto a) i).

NOTA: Leer la nota al final del archivo

Ejercicio C++
Desarrolle la clase “Medicion” para que el código “main” provisto al pie sea válido. Los objetos de la clase “Medicion” almacenan el valor de una medición en flotante y su unidad de medida como cadena de caracteres. Las mediciones se pueden operar siempre y cuando su unidad de medida sea la misma, caso contrario la operación no es permitida. También es posible, como se indica en el código al pie sumar a un flotante la “Medicion”. En este caso no es importante la unidad de medida ya que existe una sola. Use efectivamente el tiempo y no desarrolle nada que no aplique al código mostrado.
Si el código entregado contiene errores comente la línea indicando cual es el error y luego escriba la línea en forma correcta.

```
Medicion m1mv(100.0, "Mv");//millivolts
Medicion m2mv(20.0, "Mv");
Medicion m4amp(3.0, "Amp");
Medicion m3mv=m1mv-m2mv;
cout<<"Resultado 1: "<<180.0+m3mv<<endl;

try{
    cout<<m3mv-m4amp<<endl;
}catch(...){
    cout<<"No se pueden restar mediciones de distinta unidad de medida"<<endl;
}
```

EVALUACIÓN

Condición de aprobación
para obtener un 4 en el examen final debe realzar:

- EJERCICIO C:
  - crear el lote de prueba
  - realizar el “punto a” completo con su correspondiente main.
- EJERCICIO C++
  - crear la clase solicitada desarrollando el archivo “.h” completo, creando solo lo mínimo indispensable para que el main funcione correctamente. No olvide poner atención en el uso de los “const”.
- AMBOS EJERCICIOS: deben entregarse con 0 errores, 0 warnings y funcionando correctamente.

- Desarrolle cada ejercicio en un proyecto separado.
- **Incluya en el encabezado de cada archivo, // apellido_nombre_DNI**
- Recuerde antes de comprimir, eliminar las carpetas bin y obj de cada proyecto.
- **Entregue ambos proyectos compactados en un zip, “apellido_nombre_DNI.zip”.**
- Entregue el parcial usando prácticas de MIEL.
- Enviar a todos los tutores.
- ¡La evaluación es individual!

¡El mayor de los éxitos!