ESTUDIOS GENERALES CIENCIAS

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

TÉCNICAS DE PROGRAMACIÓN

EXAMEN FINAL

SEMESTRE ACADÉMICO 2025-1

Horarios: Todos

Duración: 170 minutos

Elaborado por los profesores del curso.

ADVERTENCIAS:

- Todo dispositivo electrónico (teléfono, tableta, computadora u otro) deberá permanecer apagado durante la evaluación **en su mochila**.
- Coloque todo aquello que no sean útiles de uso autorizado durante la evaluación en la parte delantera del aula, por ejemplo, mochila, maletín, cartera o similar, y procure que contenga todas sus propiedades. La apropiada identificación de las pertenencias es su responsabilidad.
- Si se detecta omisión a los dos puntos anteriores, la evaluación será considerada nula y podrá conllevar el inicio de un procedimiento disciplinario en determinados casos.
- Es su responsabilidad tomar las precauciones necesarias para no requerir la utilización de servicios higiénicos: durante la evaluación, no podrá acceder a ellos, de tener alguna emergencia comunicárselo a su jefe de práctica.
- Quienes deseen retirarse del aula y dar por concluida su evaluación no lo podrán hacer dentro de la primera mitad del tiempo de duración destinado a ella.

INDICACIONES:

- No se pueden usar apuntes de clase ni calculadoras.
- Está prohibido el acceso a Internet y a correo electrónico hasta que lo indiquen los jefes de práctica, tampoco podrá emplear dispositivos USB.
- Si se detecta omisión al punto anterior, la evaluación será considerada nula y podrá conllevar el inicio de un procedimiento disciplinario en determinados casos.
- LAS SOLUCIONES DEBERÁN DESARROLLARSE BAJO UN ESTRICTO DISEÑO DESCENDENTE, por lo que NO SE CALIFICARÁN aquellos módulos que son llamados por otros que estén incompletos. Cada módulo no debe sobrepasar las 30 líneas de código aproximadamente.
- NO SE PUEDEN EMPLEAR ARCHIVOS DE DATOS AUXILIARES NI VARIABLES GLOBALES. NO podrá implementar funciones en el archivo main.cpp, las funciones se deberán implementar en archivos independientes (.h y .cpp).
- En la calificación se tomará en cuenta el buen uso de los nombres de los identificadores, y el eficaz uso de comentarios.
- **DEBE COLOCAR SU NOMBRE Y CÓDIGO EN EL ARCHIVO main.cpp DE SU PROYECTO, DE LO CONTRARIO SE LE DESCONTARÁ 0.5 PUNTOS EN SU NOTA FINAL. NO SE HARÁN EXCEPCIONES.**
- **DEBE COLOCAR UN COMENTARIO AL INICIO DEL ARCHIVO main.cpp CON UNA DESCRIPCIÓN DE LO QUE HACE EL PROGRAMA. ESTA DESCRIPCIÓN NO DEBE SER GENÉRICA, DEBE SER ESPECÍFICA DE LO QUE HARÁ EL PROGRAMA. SE LE DESCONTARÁ 0.5 PUNTOS EN SU NOTA FINAL SI NO SE COLOCA ESTE COMENTARIO O LO QUE SE EXPRESE EN ÉL NO SEA ESPECÍFICO. NO SE HARÁN EXCEPCIONES.**
- **CADA ESTRUCTURA DEBE DEFINIRSE EN UN ARCHIVO .h INDEPENDIENTE, DE NO RESPETAR ESTE REQUERIMIENTO, NO SE ASIGNARÁ PUNTAJE EN LA DECLARACIÓN DE LAS ESTRUCTURAS.**
- **No se calificará el código puesto como comentario, ni aquellas funciones que no son invocadas en el proyecto o que su invocación esté comentada.**
- **En la calificación se tomará en cuenta el buen uso de los nombres de los identificadores, y el eficaz uso de comentarios.**
- **NO SE PERMITIRÁ LA LECTURA DE DATOS CARÁCTER POR CARÁCTER.**
- **TODA OPERACIÓN DE BÚSQUEDA DEBE REALIZARSE EN FUNCIONES INDEPENDIENTES Y DEBE CONSIDERAR QUE EL DATO BUSCADO NO SE ENCUENTRE.**

INDICACIONES INICIALES

Siga estrictamente las indicaciones que a continuación se detallan:

En la unidad de trabajo indicada por los Jefes de práctica (**si trabaja en otra unidad, no se calificará su examen y se le asignará como nota cero**), cree allí una carpeta con el nombre **"Examen2_2025_1_CO_PA_PN"** donde **CO** indica: Código del alumno, **PA** indica: Primer Apellido del alumno y **PN** primer nombre (de no colocar este requerimiento se le descontará 2 puntos de la nota final). **Allí colocará los proyectos solicitados en esta prueba. NO SE HARÁN EXCEPCIONES SI NO ACATA ESTAS INDICACIONES.**

DEBE LEER TODA LA PRUEBA ANTES DE EMPEZAR A DESARROLLAR EL PROGRAMA INCLUYENDO LAS INDICACIONES Y ADVERTENCIAS

Para establecer un **nuevo plan de fiscalización del tránsito terrestre y aéreo**, una municipalidad requiere realizar un análisis integral que considere tanto las infracciones de tránsito cometidas por los conductores como los reportes generados por la flota de drones de vigilancia. El área de **Gestión de la Información** ha brindado **cuatro archivos** que pueden ser utilizados en dos aplicaciones independientes que procesen cada conjunto de datos por separado:

La **primera aplicación**, desarrollada en lenguaje C++, debe procesar tres archivos principales: el listado de conductores habilitados que contiene los datos personales y la licencia correspondiente (**conductores.csv**), la

información de los vehículos registrados con el detalle de la placa y el modelo (**vehiculos.csv**) y el historial de infracciones de tránsito cometidas durante el periodo de análisis, que incluye la fecha, la gravedad y el monto de las multas (**infracciones.csv**). Esta aplicación debe ser capaz de asociar cada infracción a su conductor y su vehículo, calcular los montos finales de multa según la gravedad de la falta y generar documentos individuales con el formato requerido para la notificación de las sanciones.

La **segunda aplicación**, también implementada en C++, tiene como objetivo procesar la información capturada por la red de drones de vigilancia, a partir del archivo de infracciones registrados por los drones desplegados (**infracciones_drones.csv**), el cual contiene el detalle de las actividades detectadas y la clasificación de cada infracción según su gravedad. Esta aplicación debe permitir procesar los registros de infracciones, gravedad de la infracción y generar un reporte que muestre la actividad de los drones.

Cada aplicación debe desarrollarse como un proyecto independiente en C++, asegurando el manejo correcto de archivos, la validación de datos y la generación de resultados en el formato especificado por la municipalidad.

Archivo: conductores.csv

```
95822412,Lucía,Garcia,C5506
42868828,Claudia,Martinez,C2679
83197857,Pedro,Herrera,B1520
13999315,Pedro,Ramirez,A9279
…        …     …       …
```

Archivo: vehiculos.csv

```
10076758,GCU985,Fiat Uno
10226999,QTN950,Toyota Corolla
10435578,FSS165,Renault Logan
10872248,BVQ944,Peugeot 208
…        …      …
```

Archivo: infracciones.csv

```
24549543,17-06-2025,B2,NO RESPETAR SENAL,Muy Grave,200.0
45264581,09-06-2025,E5,HABLAR CELULAR,Leve,300.0
13999315,20-06-2025,A1,LUZ ROJA,Muy Grave,300.0
…        …          …  …          …     …
```

Archivo: infracciones_drones.csv

```
PK4-003,Grave,234,Conducir sin habilitacion
IP6-005,Leve,104,Luces apagadas
IP6-005,Leve,108,Obstruir paso peatonal
IP6-005,Grave,222,Transportar explosivos
PK4-003,Leve,130,No usar luces intermitentes
…
```

En el archivo **"conductores.csv"**, se ha colocado en cada línea la información de un conductor registrado. Por cada línea del archivo se cuenta con el **DNI** del conductor (un número entero de 8 dígitos), el **nombre**, el **apellido** y la **licencia** de conducción que posee.

En el archivo **"vehiculos.csv"**, se ha colocado la información de los vehículos registrados en la base de datos. Cada línea contiene el **DNI** del propietario del vehículo, la **placa** del vehículo y el **modelo o marca** de este. Este archivo incluye tanto vehículos asociados a conductores que figuran en el archivo de conductores como también vehículos no asociados a ningún conductor registrado.

En el archivo **"infracciones.csv"**, se ha colocado el detalle de las infracciones registradas durante el periodo considerado. Por cada línea se cuenta con el **DNI** del conductor que cometió la infracción, la **fecha** en que se cometió (en formato *dd-mm-aaaa*), el **código** de la infracción, una **descripción breve** de la falta cometida, el nivel de **gravedad** de la infracción (puede ser "Leve", "Grave" o "Muy Grave") y el **monto** de la multa impuesta en soles. El archivo está ordenado cronológicamente por las fechas.

En el archivo **"infracciones_drones.csv"**, se ha colocado el registro de infracciones detectadas por los drones de vigilancia durante sus patrullajes. Cada línea contiene el **identificador del dron** que detectó la infracción, el nivel de **gravedad** asignado al incidente, el **código del incidente** y una breve **descripción de la infracción**. Dado que un mismo dron puede detectar múltiples infracciones, su identificador puede aparecer repetidas veces en el archivo, lo que refleja que un único dron puede tener varias infracciones asociadas.

PREGUNTA 1 (10 puntos): EN ESTA PREGUNTA SERÁ OBLIGATORIO EL USO DE ESTRUCTURAS Y ARREGLOS

Cree un proyecto en NetBeans con el nombre: "Pregunta1_2025_1" (de no respetar este nombre se le descontarán dos puntos de su nota final – NO SE HARÁN EXCEPCIONES) y en él desarrolle el programa que resuelva el problema que se describe a continuación. El proyecto debe estar dentro de la carpeta creada previamente y debe contener todas las funciones necesarias para su correcto funcionamiento.

CONSIDERACIONES

a) NO DEBE HACER QUE DOS O MÁS PUNTEROS APUNTEN AL MISMO DATO.
b) No debe repetir código innecesariamente, tareas como la lectura y escritura de los textos deben ser realizadas en una única función que se adapte a todas las situaciones del problema.
c) Será parte importante de la nota el formato del reporte, éste deberá ser lo más parecido a la muestra dada. En este sentido, todos los valores deben estar correctamente alineados. No se podrá emplear el carácter de tabulación ('\t') para la emisión del reporte.

Las tareas por realizar son las siguientes, las cuales debe desarrollarlas en el orden que se indican y cada una en una función independiente:

a) (1.0 punto) Definir las siguientes estructuras:

   Fecha: que debe contener los siguientes campos: 1) **aa** (valor entero del año), 2) **mm** (valor entero del mes), 3) **dd** (valor entero del día) y 4) **aammdd** (valor entero que representa la fecha completa).
   Vehiculo: que debe contener los siguientes campos: 1) **dni_propietario** (valor entero), 2) **placa** (cadena de caracteres dinámica) y 3) **marca_modelo** (cadena de caracteres dinámica).
   Infraccion: que debe contener los siguientes campos: 1) **fecha_infraccion** (estructura **Fecha**), 2) **codigo** (cadena de caracteres dinámica), 3) **descripcion** (cadena de caracteres dinámica), 4) **gravedad** (cadena de caracteres dinámica), 5) **monto** (valor de punto flotante) y 6) **dni_conductor** (valor entero).
   Conductor: que debe contener los siguientes campos: 1) **dni** (valor entero), 2) **nombre** (cadena de caracteres dinámica), 3) **apellido** (cadena de caracteres dinámica), 4) **licencia** (cadena de caracteres dinámica), 5) **vehiculo** (estructura **Vehiculo**), 6) **infracciones** (arreglo dinámico de **Infraccion**) y 7) **cantidad_infracciones** (valor entero). Considere máximo 10 elementos en el arreglo dinámico de infracciones.

b) (1.0 punto) Leer los datos del archivo **conductores.csv** y alojarlos en una variable llamada conductores que es un arreglo dinámico de estructura **Conductor** de no más de 120 elementos.

c) (1.0 punto) Leer los datos del archivo **vehiculos.csv** y alojarlos en una variable llamada **vehiculos** que es un arreglo dinámico de estructura **Vehiculo** de no más de 160 elementos.

d) (2.0 puntos) Leer los datos del archivo **infracciones.csv** y alojarlos en una variable llamada infracciones que es un arreglo dinámico de estructura **Infraccion** de no más de 160 elementos. Para cada infracción deberá procesar la fecha para llenar correctamente la estructura **Fecha**. Si la infracción posee un código de tipo **"CX"**, donde **X** es un número entero, el monto de la infracción deberá incrementarse en un 20%.

e) (3.5 puntos) Actualizar la información de los conductores y sus infracciones: para cada conductor deberá recorrer los arreglos de vehículos e infracciones y asociar los datos correspondientes. Primero, deberá buscar y asignar el vehículo que le pertenece según su DNI. Luego, deberá identificar todas las infracciones cometidas por ese conductor colocarlas en su arreglo dinámico de infracciones.

f) (1.5 puntos) Emitir las papeletas correspondientes: **Cree una carpeta llamada "papeletas" en el directorio raíz de su proyecto.** Por cada infracción registrada de cada conductor, deberá generar un archivo de texto cuyo nombre se construye concatenando el apellido, el nombre y la licencia del conductor, por ejemplo: **CastroPaulaC7543.txt**. El archivo deberá guardarse dentro de la carpeta creada. Cada papeleta debe contener la fecha de la infracción, los datos completos del conductor, los datos del vehículo, la descripción y la gravedad de la infracción, así como el monto final a pagar. El contenido de cada papeleta debe presentarse con un formato claro y ordenado que facilite su lectura, respetando la estructura indicada por el enunciado. En el siguiente cuadro se muestra un ejemplo del contenido del archivo de texto para **una papeleta** creada:

"papeletas/CastroPaulaC7543.txt" *(debe generar una cadena con este formato para dar el nombre al archivo)*

```
Fecha de Infraccion: 15/05/2025

Conductor:
  Nombres       : Paula
  Apellidos     : Castro
  DNI           : 45264581
  Licencia      : C7543

Vehiculo:
  Placa         : ELX698
  Marca/Modelo  : Ford Fiesta

Infraccion:
  Código        : C3
  Descripción   : SIN CINTURON
  Gravedad      : Leve

Monto a Pagar: S/ 350.00
```

PREGUNTA 2 (10 puntos): EN ESTA PREGUNTA SERÁ OBLIGATORIO EL USO DE ESTRUCTURAS AUTOREFERENCIADAS.

Cree un proyecto en NetBeans con el nombre: "Pregunta2_2025_1" (de no respetar este nombre se le descontarán dos puntos de su nota final – NO SE HARÁN EXCEPCIONES) y en él desarrolle el programa que resuelva el problema que se describe a continuación. El proyecto debe estar dentro de la carpeta creada previamente y debe contener todas las funciones necesarias para su correcto funcionamiento.

La municipalidad en mención ha iniciado la fase de prueba de un sistema de patrullaje aéreo con drones para registrar infracciones de tránsito. Cada infracción detectada puede clasificarse como **Leve** o **Grave**, y según su gravedad se almacenará en una **lista simplemente ligada** siguiendo estas reglas:

- Las infracciones **Graves** se insertan al **inicio** de la lista.
- Las infracciones **Leves** se insertan al **final** de la lista.

Este mecanismo permite mantener un orden tal que todas las infracciones graves queden al principio de la lista, facilitando su atención prioritaria, mientras que las infracciones leves se vayan acumulando al final para un procesamiento posterior.

CONSIDERACIONES

- No puede desarrollar una tarea si no ha desarrollado por completo la tarea anterior, de hacerlo no se corregirá esa tarea.
- Será parte importante de la nota el formato del reporte, éste deberá ser lo más parecido a la muestra dada. En este sentido, todos los valores deben estar correctamente alineados. No se podrá emplear el carácter de tabulación ('\t') para la emisión del reporte.
- Toda operación de búsqueda debe realizarse en una función independiente. No se considerará en la calificación los procesos de búsqueda que estén contenidos en el código de otro proceso. Las funciones de búsqueda deben considerar la posibilidad que el dato buscado no se encuentre.

Las tareas por realizar son las siguientes, las cuales debe desarrollarlas en el orden que se indican y cada una en una función independiente:

a) (1.0 punto) Definir las siguientes estructuras:

Infraccion: que debe contener los siguientes campos: 1) **codigo** (valor entero), 2) **descripcion** (cadena de caracteres dinámica) y 3) **gravedad** (cadena de caracteres dinámica).
Nodo: que debe contener los siguientes campos: 1) **infraccion** (estructura **Infraccion**) y 2) **siguiente** (puntero a **Nodo**). Esta es una estructura autoreferenciada.

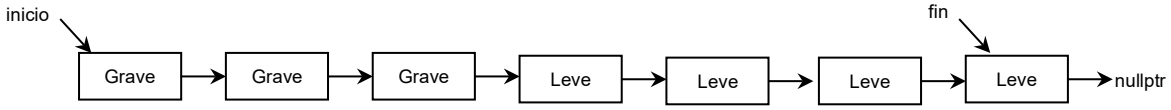
Para implementar la lista de **Nodo** puede decidir entre uno de los siguientes esquemas:

Esquema 1: Crear la estructura **Lista**: que debe contener los siguientes campos: 1) **inicio** (puntero a Nodo), 2) **fin** (puntero a Nodo). Esta es un **TAD**.

Esquema 2: Definir la lista usando dos punteros: 1) **inicio** (puntero a Nodo), 2) **fin** (puntero a Nodo).

b) (1.0 punto) Cree dos listas para guardar la información de un dron cada una. Para esto deberá leer desde el teclado la identificación de los dos drones con los que desea trabajar (por ejemplo: IP6-005 y PK4-003), los drones que no correspondan a estos códigos se deben descartar. Asuma que los datos ingresados por el usuario son correctos.

c) (5.0 puntos) Para ambas listas completar las infracciones del dron que le corresponda. En ambos casos debe leer los datos del archivo **infracciones_drones.csv** y colocar cada infracción en la lista de infracciones del dron correspondiente. Si una infracción es de gravedad "Leve", deberá insertarse al final de la lista simplemente ligada; si la infracción es de gravedad "Grave", deberá insertarse al inicio de la lista simplemente ligada de infracciones. Tenga en cuenta que solo debe usar una función para completar la lista y esta función debe ser invocada para cada dron. No se requiere que las listas estén ordenadas por algún criterio. No debe recorrer toda la lista para insertar el nodo al final de la lista, debe usar el puntero fin.



d) (3.0 puntos) Emitir un reporte con la información cargada de los drones, el reporte debe lucir de esta manera:

reporte_drones.txt

```
=== Dron IP6-005 ===
  [Grave] Codigo: 209 - Evadir control policial
  [Grave] Codigo: 208 - Conducir con carga peligrosa
  …
  [Leve] Codigo: 108 - Obstruir paso peatonal
  [Leve] Codigo: 134 - Frenar innecesariamente
  …
  Leves: 24, Graves: 18
====================
=== Dron PK4-003====
…
```

Al finalizar el examen, comprima la carpeta que contiene los dos proyectos empleando el programa Zip que viene por defecto en el Windows, no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares y súbalo a la tarea programada en PAIDEIA para este examen.

ADVERTENCIAS:
Obligatoriamente debe desarrollar sus proyectos bajo NetBeans en Windows, no podrá desarrollarlo empleando otro IDE ni otro sistema operativo.

CRITERIOS DE CALIFICACIÓN:
1. Si el programa entregado presenta más de tres errores de sintaxis serán calificados sobre la mitad del puntaje. 2. Si el programa no muestra los resultados o los muestren y no sean correctos, no podrán tener más del 75% de la nota. 3. Se descontará 15% de la nota si el programa define variables con nombres que no tengan sentido. Las variables deben empezar con una minúscula, se emplearán mayúsculas o guiones para separar las palabras compuestas (p. e.: baseInferior). 4. Se descontará 15% de la nota si no se colocan comentarios relevantes, 5. No se calificarán aquellas funciones implementadas en el archivo main.cpp

San Miguel, 07 de julio del 2025