**II Examen Parcial**

Nombre estudiante: Nicole de los Ángeles Araya Ballestero\_\_\_\_\_ Id \_4 0250 0623\_\_\_\_\_

**Indicaciones:**

* **El examen inicia a las 2:00 pm de este domingo 24 de octubre y se deberá entregar a más tardar antes de las 8:00 pm**
* Se entiende que el examen es estrictamente individual, si se demuestra plagio, copia parcial o total, se procederá inmediatamente a su anulación, la nota del examen será cero (0) y se realizará comunicado a las instancias respectiva, donde será(n) sancionado(s) de acuerdo con el Reglamento General sobre los Procesos de Enseñanza y Aprendizaje de la Universidad Nacional.
* Tome en cuenta que el examen es programado, a “libro abierto” y en la comodidad de su hogar, puede contar con todo el material de apoyo que usted ha recibido de su profesor y que se encuentra en el aula virtual.
* Para ahorrar un poco de tiempo, se supone que los estudiantes tendrán algunas clases previamente listas y realizadas, tales como una contenedora simple, entre otras.
* Dada esta modalidad de examen, se espera que cada ítem desarrollado funcione adecuadamente, en caso de no funcionar será calificado con un **cero**, esto dada la naturaleza misma del examen, ya que el estudiante tiene a su disposición todos los recursos, por lo cual no se evalúa memoria o retención, sino uso correcto y funcional de las herramientas, así como las destrezas y habilidades en el uso de las herramientas.
* El examen deberá ser entregado en un proyecto completo realizado en el IDE oficial definido por cada profesor, dicho proyecto deberá estar completo y compilar sin errores y además ser funcional resolviendo de forma adecuada lo solicitado.
* El examen se deberá entregar vía aula virtual a más tardar antes de las **8:00pm**. Una vez finalizado el tiempo establecido no se recibirán entregas tardías No se reciben exámenes vía correo, salvo en caso eventual que la plataforma institucional, estuviera caída. Verifique que el proyecto entregado en el aula virtual sea la versión final de su examen, ya que posteriormente no se aceptarán reclamos por envío de archivos equivocados.
* Estudiantes con “**Ajustes Metodológicos**” (antes conocido como adecuación curricular), cuentan con tiempo adicional **(2.0 horas más)**. Cada profesor definirá el modo de entrega para estos casos.
* La forma de entrega es en un archivo con el siguiente formato: #ced.zip o #ced.rar. Debe asegurarse que el archivo no esté corrupto.

**Parte Única**

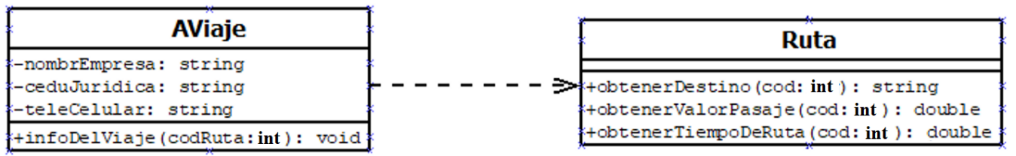
A continuación, se presentan cuatro situaciones a resolver bajo el paradigma orientado a objetos, por parte del estudiante, si desean pueden utilizar la programación base que se les ofrece en el examen. **Si desean pueden no utilizarla**. **Tampoco es obligatoria utilizarla**. Los ejercicios están pensados con base en lo visto en las clases virtuales. Los objetos para crear, deberán ser dinámicos.

**Problema No.1 (Tema: Relación UML)**

La Empresa de buses “Viajes Costa Rica”, desea un pequeño programa que le resuelva la siguiente situación. Implementar dos clases una llamada la clase AViaje con atributos todos de tipo string, que son: nombrEmpresa, ceduJuridica, teleCelular, esta clase depende de otra (que solo tiene métodos), llamada clase Ruta, que le ayuda a mantener información que se observa en la siguiente tabla:



Se observa, en la tabla, que teniendo el código del viaje se tiene gran información de ese viaje. Implemente esa clase Ruta (que es una clase de servicio), para ser utilizada en la clase AViaje con base en la siguiente UML. La clase AViaje tiene un método llamado void infoDelViaje(int) que dependiendo del código recibido, genera toda la información al respecto a esa ruta. Realizar un main( ) que ejemplifique el uso de esta relación por lo menos utilizando dos códigos. Puede utilizar el siguiente UML.



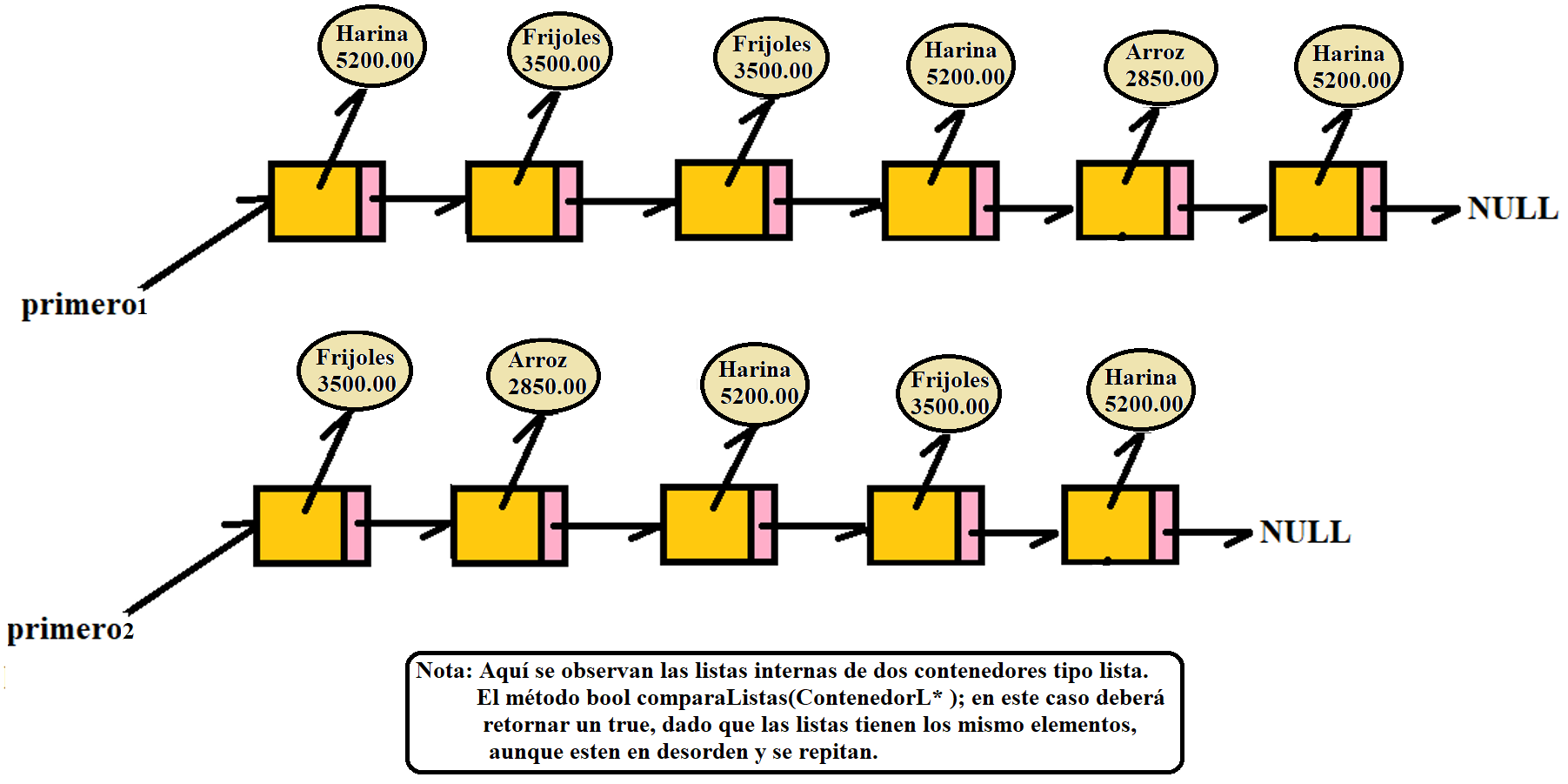
**Puntaje para el problema No.1**

1. Creación y funcionalidad correcta de los tres métodos de la clase Ruta. (10 pts).
2. Creación y funcionalidad correcta del método infoDelViaje(int). (10 pts)
3. Creación y funcionalidad de un pequeño main que ejemplifique esa relación (5 pts).

**Problema No.2 (Tema: Listas)**

En este problema se tiene ya implementada la clase Nodo y la clase Lista (Contenedor tipo Lista), que guarda o almacena objetos de tipo Producto. La clase Producto tiene solo dos atributos, de tipo string: nombreP y otro double precioP. Realizar un método para la clase Lista, llamado: bool compararaListas(ContenedorL\*); que recibe objetos dinámicos tipo ContenedorL, y compara la lista (this) con el contenedor tipo lista que llega por parámetro. El método nos garantiza que los elementos que están en una lista, existen en la otra, aunque el orden de los elementos no sea el mismo y en cada lista pueden existir elementos repetidos.

***Ejemplo:*** a) En la siguiente figura se dibujan dos listas de dos contenedores tipo Lista diferentes, donde cada nodo tiene su respectivo objeto dinámico tipo Producto.



**Nota No1**. El dibujo solo es un ejemplo de las listas internas de los dos contenedores tipo Lista. Los punteros tipo Nodo\* primero1 como primero2, solo indica que se trata de dos listas diferentes.

**Nota No2.** La comparación es verificar si los contenedores tipo listas son iguales, es decir que todo objeto dinámico producto de una lista se debe repetir, por lo menos una vez en la otra lista.

**Nota No3.** Tener cuidado si se repite el nombre del producto, pero con otro precio y este no se encuentre en la otra lista. En tal caso el método deberá retornar un false.

**Puntaje para el problema No.2**

1. Escritura correcta de las clases Producto, Nodo y ContenedorL con su (.h) y (.cpp) (3 pts).
2. Escritura correcta del método bool esIgualA(Producto&) de la clase Producto. (5 pts).
3. Escritura y funcionalidad del método bool comparaListasIguales(ContenedorL\*) en la clase ContenedorL a la hora de establecer si los contenedores son o no iguales. (12 pts).
4. Escritura correcta de un main que verifique la funcionalidad de esos métodos (5 pts).

**Problema No.3 (Tema: Matrices)**

Un hotel (Ocean Drive Madrid) de 4 pisos y 6 habitaciones por piso, lo ha contratado a usted para que le programe llevar el control de las habitaciones del hotel y de lo recaudado por día. A usted como programador se le ocurrió que podría implementar un Contenedor Matricial para resolver esta situación y así escribir una clase ContenedorM que almacene objetos dinámicos tipo Habitación. Es decir, poder construir un Contenedor Matricial de 4 filas y 6 columnas para ubicar un objeto Habitación en cada casilla de la matriz. La clase Habitación debe tener atributos básicos tales como: numHabitac:int, numCamasAdult:int, numCamasNinos:int estado:char, bool:desc; que se refiere al estado de la habitación. La habitación puede tener uno de tres estados posibles (‘O’ = Ocupado, ‘D’ = Desocupado, ‘M’ = Mantenimiento).

Cada vez que se alquila una habitación (en la recepción) se clasifica a las personas que la habitarán. Si al alquilar una habitación, se clasifica a las personas con un **desc** tipo **true**, entonces a las camas de los niños se les hace un descuento de un 50% independientemente de cuantos niños lleguen, obviamente no pueden exceder el cupo que hay para niños en dicha habitación. Pero generalmente se clasifican a las personas que habitarán la habitación con un **desc** tipo **false. Nota 1: Ocupada = Alquilada.**

En cada habitación, cada cama para niños tiene un costo de 15 mil diarios, cada cama para adulto tiene un costo de 20 mil diarios. En toda habitación hay dos camas para adultos y 3 camas para niños. Construya un método de la clase ContenedorL llamado: double obtenerRecaudoDiarioTotal( ), que dependiendo del estado de cada habitación, retorne la cantidad de dinero que se ha recaudado un día en particular, dependiendo del aforo del hotel. Es decir, saber que si la habitación está desocupada o en mantenimiento no generarán ganancia ese día.

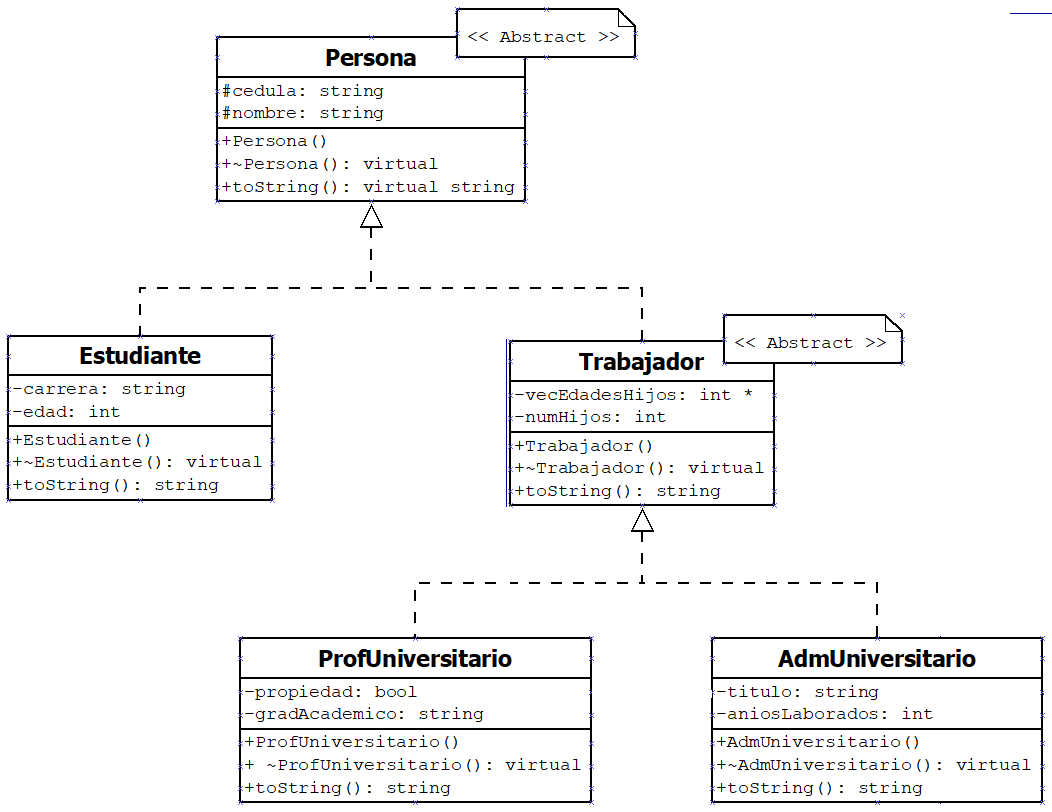
**Nota 2**. Cada habitación es alquilada como si todas sus camas se fueran a utilizar, es decir, si solo llega una persona a alquilar una habitación, deberá pagar por todas las camas restantes de la habitación como si se fueran a utilizar. **Nota 3**: Observe que solo hay un máximo de 24 habitaciones.

**Puntaje para el problema No.3**

1. Escritura correcta de las clases Habitación, ContenedorM, OO con sus (.h) y (.cpp) (5 pts).
2. Correcta funcionalidad cuando se debe establecer el estado de una habitación ya sea que se habilite para desocupada, en mantenimiento o ocupada. (9 pts).
3. Impresión de la matriz de habitaciones con su número y estado D-M-O.(6 pts).
4. Cálculo e Impresión de la recaudación total de dinero del hotel por día(5 pts).
5. Se da un main casi terminado con su respetivo Menú para ayudar la app. (sin puntos.).

**Problema No.4 (Tema: Herencia)**

****A continuación, se presenta un UML de una herencia. Se trata de una jerarquía de clases de personas universitarias donde se observan que existen estudiantes, profesores y administrativos. También se observa que la clase Persona es abstracta, es decir que tiene como mínimo un método virtual puro, que en este caso es el toString(). Así que el prototipo de ese método (función miembro), debería ser en C++ igual a: **virtual string toString( ) = 0;**



**Nota 1**: Las líneas punteadas de la herencia significan que se trata de una realización, es decir una herencia donde se está heredando de una clase abstracta. Así que, las clases Persona y Trabajador son clases Abstractas dado que su método toString() es un método virtual puro (= 0).

**Puntaje para el problema No.4**

1. Escriba las clases de la herencia (.h), (.cpp) en el editor de texto del compilador. (4 pts).
2. Escribir un programa en el main, donde se puedan crear tres objetos dinámicos (Estudiante, ProfUniversitario y AdmUniversitario) donde sus atributos (y las edades de sus hijos en el caso de los Trabajadores), puedan ser llenados, desde el teclado. (15 pts).
3. Una vez creado esos objetos se deben de imprimir correctamente toda su información con el uso del método toString( ). (6 pts).