Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

8 de junio de 2018

Examen de **Tecnología de Programación**

Grado en Ingeniería Informática

Duración total del examen: **2 horas 30 minutos**

NOTA RECORDATORIA. La evaluación final consta de las siguientes pruebas:

- Prueba escrita → 60% de la nota (mínimo 5 para aprobar)
- Prácticas (máxima nota entre prácticas entregadas durante el curso y el examen de prácticas) → 40% de la nota

Ejercicio 1 [2 puntos]

A continuación verás dos programas realizados en el lenguaje C++. Por cada uno de los programas podrá haber más de un fichero cuyo nombre aparece encima de su código. Todos los ficheros están en la misma carpeta. De cada uno de los programas, responde a las siguientes preguntas:

- Se compila con el comando `g++ main.cc -std=c++11` (gcc versión $>$ 4.8.1). ¿Compilaría correctamente o daría algún error? Ignora errores tipográficos y potenciales *memory leaks*.
- Si tu respuesta es que no compila ¿dónde ocurre el error de compilación? ¿por qué no compila?
- Si tu primera respuesta es que sí que compila ¿qué sacaría el ejecutable por pantalla? No lo justifiques.

Programa A

foo.h

```
class Foo {
protected:
 double f;

public:
 Foo(double f) :
    f(f) {}

 void sil()
 {  f/=2.0;    }

 virtual float tolo()
                  const
 {  return f;  }
};
```

bar.h

```
#include "foo.h"
class Bar :public Foo
{
public:
  Bar(double t) :
    Foo(t+1.0) { }

  void sil()
  {  f*=2.0f;    }

  float tolo() const
              override
  {  return f-1.0f; }
};
```

main.cc

```
#include <iostream>
#include "bar.h"
using namespace std;
int main() {
  Foo* foo = new Foo(2.0);
  foo->sil();
  cout<<foo->tolo()<<endl;

  Bar* bar = new Bar(4.0);
  bar->sil();
  cout<<bar->tolo()<<endl;

  Foo* foobar = new Bar(8.0);
  foobar->sil();
  cout<<foobar->tolo()<<endl;
}
```

Programa B

foo.h

```
template<typename T>
class Foo {
protected:
 T t;

public:
 Foo(const T& t) :
    t(t) { }

 virtual T meta()
                const
 {  return t;   }
};
```

bar.h

```
#include "foo.h"
class Bar :
     public Foo<int> {

public:
  Bar() :
     Foo<int>(0) {}

  double meta() const
                 override
  {  return 1.0;   }
};
```

main.cc

```
#include "bar.h"
#include <iostream>
using namespace std;
int main() {
  Foo<int> ifoo(155);
  cout<<ifoo.meta()<<endl;

  Foo<double> dfoo(42);
  cout<<dfoo.meta()<<endl;

  Bar bar;
  cout<<bar.meta()<<endl;
}
```

Ejercicio 2 [3 puntos]

La **herencia** es un mecanismo de los lenguajes orientados a objetos que permite establecer relaciones entre diferentes clases. Una de las consecuencias de la herencia es la posibilidad de definir **clases abstractas**. Sobre este tema, realiza las siguientes tareas:

- **(a)** **Define**, con tus propias palabras y de forma independiente a un lenguaje de programación específico, el concepto de clase abstacta.
- **(b)** Explica **para qué sirve** (para qué puede utilizarse) una clase abstracta.
- **(c)** Una clase abstracta tiene una **limitación** de uso que una clase no abstracta no tiene. **¿Cuál es?** ¿Por qué tiene esa limitación?
- **(d)** **Ilustra** mediante un **ejemplo** en un lenguaje orientado a objetos a tu elección una clase abstracta y su uso. Dicho ejemplo deberá ser lo más pequeño posible (preferiblemente con solo dos clases involucradas) y deberá incluir un pequeño programa principal.
- **(e)** **Justifica** en tu ejemplo dónde se aprecia tu definición de **(a)** y tu explicación de para qué sirve una clase abstracta de **(b)**.
- **(f)** Escribe **una sola línea de código** que, puesta en el programa principal de tu ejemplo en **(d)** no compile debido a la limitación que has explicado en el apartado **(c)**. Explica por qué dicha línea de código no compilaría.

Ejercicio 3 [2 puntos]

En un futuro no distópico todos los ciudadanos llevamos un **chip subcutáneo** que permite tenernos localizados en todo momento mediante tecnología GPS, aunque a nadie parece importarle porque así pueden saber cuántos pasos dan al día o localizar los restaurantes más cercanos. El chip subcutáneo guarda diariamente para cada ciudadano toda la información del recorrido que ha realizado en una lista de tuplas. Cada tupla es de la forma `(x,y,t)` (todo números reales) donde:

- `x` e `y` son las coordenadas en metros sobre la superficie de la ciudad.
- `t` es el instante temporal en segundos desde las 00:00 del día.

Se puede asumir que todas las tuplas de la lista están ordenadas por instante temporal `t`. El ayuntamiento de Zaragoza necesita procesar los ficheros de todos sus ciudadanos para extraer cierta información y te ha encargado a ti las siguientes funciones en Haskell:

- `distancia_total recorrido` - mide la **distancia total** del `recorrido`. Se considera que entre dos tuplas el ciudadano ha ido en línea recta (distancia euclídea).
  Ejemplo :

```
*Main> distancia_total [(3,4,10),(0,0,20),(3,4,35)]
10.0
```

- `es_subversivo recorrido` - indica si el `recorrido` se corresponde (o no) con el de un ciudadano **subversivo**. Un ciudadano es subversivo si ha conseguido esconder información (alterando el chip o interceptando su señal) de tal forma que no ha guardado un paso en diez minutos desde el paso anterior o en diez minutos desde el inicio del día a las 00:00.
  Ejemplos :

```
*Main> es_subversivo [(3,4,10),(0,0,20),(3,4,35)]
False
*Main> es_subversivo [(3,4,10),(0,0,20),(3,4,3500)]
True
*Main> es_subversivo [(3,4,1000),(0,0,1010),(3,4,1035)]
True
```

- `velocidad_media varios_recorridos` - calcula la **velocidad media** de todos los recorridos en la lista de recorridos `varios_recorridos`. Dado un recorrido, su velocidad media puede calcularse globalmente con respecto a la distancia total y al tiempo total. Se considera que un recorrido que todavía no ha empezado (vacío) o un recorrido estático (que no se mueve) tiene velocidad media 0.
  Ejemplo :

```
*Main> velocidad_media [[(3,4,10),(0,0,20)],[(3,4,20)],
                                        [(3,4,10),(0,0,12),(3,4,20)]]
0.5
```

Se valorará positivamente la **brevedad y concisión** de las soluciones propuestas. Se puede utilizar cualquier función de Haskell, y también se pueden definir otras funciones propias si se considera necesario.

Ejercicio 4 [3 puntos]

La app de mensajeria **TurraApp** te ha encargado el diseño de su biblioteca de gestion de mensajes. Elige el lenguaje orientado a objetos que prefieras (Java o C++), y diseña los tipos de datos y métodos necesarios para implementar la siguiente funcionalidad:

- Pueden existir mensajes de **distintos tipos**, concretamente *Video*, *Audio*, *Foto* o *Texto*.
- **Todos** los mensajes almacenados **contienen** un identificador (entero), el nombre del emisor, y la fecha y hora en que se enviaron, y nos permiten obtener su tamaño en bytes, dependiendo de su contenido.
- Los medios audiovisuales (*Video* y *Audio*) tienen asociada una **duracion en segundos**.
- Los medios de imagen (*Video* y *Foto*), tienen asociado un **lugar** donde fueron capturados (su nombre).
- Los mensajes contienen datos especificos de su tipo (el *Video* o la *Imagen* pueden tener información de su resolución, por ejemplo). Aunque no nos preocuparemos de ellos, si que nos interesa poder obtener para todos una *Miniatura* para representarlos en la ventana de chat. El **metodo de obtencion de esa minitura** es distinto para cada tipo de mensaje.
- Todos **los mensajes se almacenan** en la *Mediateca* de la app a nivel global, y en cada uno de los *Chats* donde aparecen. Los mensajes pueden aparecer en varios chats (si se han reenviado), pero no se duplican. Los mensajes se borran al eliminar el ultimo chat que los usa.

La empresa desea automatizar la gestion de los medios, asi que te piden que implementes la siguiente funcionalidad (funciones o métodos):

- **Crear** un nuevo *Chat*.
- **Añadir** un mensaje al *Chat*.
- **Borrar** todos los mensajes anteriores a un instante concreto.
- **Generar** la lista de miniaturas para un *Chat*
- **Generar** la lista de miniaturas para todos los medios de tipo audiovisual (*Audio* o *Video*) mayores de un tamaño dado.

Puedes suponer que los tipos de datos `Miniatura`, `Instante` están predefinidos, con los metodos y/o operadores que consideres necesarios, como igualdad o comparación.