NOMBRE: _____**Titulación:**_____

ESTADO DE LAS PRÁCTICAS: _____

1.- Dado el siguiente código **Eiffel**:

```
class Consultor feature{ALL}
      nombre: String;
      fechaNacimiento: Date;
      honorarios: REAL;
      nivelExperiencia: INTEGER;
      …
end
```

a) (**0'75 ptos**) De acuerdo con el ***Principio de Ocultamiento de la Información*** ¿es conveniente definir los atributos de la clase `Consultor` en **Eiffel** como públicos? ¿y en **Java**?

b) (**0'75 ptos**) Supuesto que a los atributos `honorarios` y `nivelExperiencia` sólo se van a acceder desde la clase `Empresa` y el resto se deben poder consultar desde cualquier clase pero no modificar ¿cuál sería el nivel de visibilidad de cada uno de los atributos en **C++**? Razona la respuesta.

c) (**0'5 ptos**) ¿Qué papel juegan las propiedades en la definición de una clase en **C#**?

2.- En una empresa han definido el método **Java** que se muestra a continuación para seleccionar el primer consultor que cumple uno de los criterios de selección definidos: el primero que se ajuste al precio o el primero que tenga el nivel de experiencia exigido.

```
Consultor encuentraPrimerCandidato(List<Consultor> consultores,
                                   boolean conformePrecio,
                                   float maxPrecio,
                                   boolean conformeExperiencia,
                                   int minNivel) {
        for (Consultor consultor; consultores) {
            if (conformePrecio && consultor.getHonorarios() > maxPrecio)
               continue;

            if (conformeExperiencia && consultor.getExperiencia() < minNivel)
               continue;

            return consultor;
        }
        return null;
}
```

a) (**0'5 ptos**) ¿Favorece el código anterior el ***Principio de Abierto-Cerrado***? Justifica la respuesta.

b) (**0'5 ptos**) ¿Favorece el código anterior el criterio de reutilización de ***Variación de Algoritmos y Estructuras de Datos***? Justifica la respuesta.

c) (**1 pto**) Modifica el código anterior e implementa todo el código adicional que sea necesario para poder obtener la misma funcionalidad pasándole como parámetro la estrategia de selección de consultores. De acuerdo a la implementación propuesta, completa convenientemente la siguiente llamada:

```
Consultor consultorBarato = encuentraPrimerCandidato(consultores, XXX);
```

d) (**1 pto**) Explica el concepto de ***iterador interno*** y, a partir de la implementación de dichos iteradores vista en clase, implementa el iterador `IteradorBaja` encargado de generar un listado de los consultores que se han jubilado. NOTA: Un trabajador se jubila cuando cumple 65 años. Todo consultor tiene un método `getEdad`.

e) (**0'5 ptos**) La empresa cuenta con distintos tipos de consultores dependiendo del tema que se va a tratar, incluyendo: `ConsultorFinanciero`, `ConsultorLegal`. Implementa el código **Eiffel** equivalente al siguiente código Java:

```
public ConsultorFinanciero maxBeneficios (List<Consultor> consultores) {
      float actual,result=-1;
      ConsultorFinanciero consultorSeleccionado = null;

      for (Consultor consultor; consultores){
            if (consultor instanceof ConsultorFinanciero){
             ConsultorFinanciero financiero = (ConsultorFinanciero)consultor;
             actual = financiero.getBeneficios();
            if (actual>result) {
                result = actual;
                consultorSeleccionado = financiero;
            }
      }
      return consultorSeleccionado;
}
```

f) (**0'5 ptos**) Compara el mecanismo de redefinición de métodos de Java y C#.

g) (**0'5 ptos**) ¿Es equivalente definir la colección de consultores como `List` en lugar de `List<Consultor>`? En el caso de que sea diferente explica por qué e indica los cambios que habría que hacer en el código Java del método `maxBeneficios`.

h) (**0'5 ptos**) ¿Es equivalente definir la colección de consultores como `List<T extends Consultor>` en lugar de `List<Consultor>`? Razona la respuesta.

3.- El método **Java** `sendEmail` de la clase `Empresa` es el encargado de enviar un correo electrónico al empleado que se le pasa como parámetro. Para ello hace uso de las clases `EmailMessage` y `Transport` como se ve en el siguiente código:

```
public class Empresa{
    private String contacto = "empresa@empresa.com";
    …
    public void sendEmail(Empleado para, String asunto, String cuerpo){

      //argumentos:(String origen, String destino, String asunto, String cuerpo)

      EmailMessage mensaje = new EmailMessage(contacto,
                                              para.getEmail(), asunto, cuerpo);
      Transport.send(mensaje);
    }
}
```

a) (**0'5 ptos**) El constructor de la clase `EmailMessage` lanza una excepción si las cadenas de texto que contienen las direcciones de correo origen y destino no están bien formadas. ¿De qué tipo es está excepción? Razona la respuesta.

b) (**0'5 ptos**) El método `send` de la clase `Transport` lanza una excepción si se produce algún fallo en el envío del mensaje. ¿De qué tipo es está excepción? Razona la respuesta.

c) (**1 pto**) Modifica el código del método `sendEmail` para manejar las excepciones convenientemente de manera que el método que invoque a `sendEmail` conozca si el envío se llevó a cabo con éxito o no. Nota: en la clase `EmailMessage` existe el método de clase `isWellFormed` que recibe como argumento una cadena de texto y devuelve `true` si se corresponde con una dirección de correo bien formada o `false` en caso contrario.

4.- (**1 pto**) La tienda SweetTeeth se dedica a la venta de chucherías. Las chucherías se venden individualmente (gominola, chicle, piruleta, etc.), cada una con un precio, o se pueden preparar bolsas que contienen una selección de los productos que se venden de manera individual o también otras bolsas. Por ejemplo, una bolsa podría contener: un chicle, una piruleta y una bolsa de gominolas. El precio de una bolsa se calcula como la suma de todos los productos que contiene más 25 céntimos por el coste de preparación. Representa gráficamente la jerarquía de clases con **herencia múltiple** que caracteriza los productos de la empresa SweetTeeth e implementa **en Eiffel** el método `getPrecio` en cada una de las clases.