Teórico. 30 %

1. La firma del método *setter* para el atributo tipo *string matriculaAlumno* de la clase ***Alumno*** es:
    a. string setMatricula();
    b. void setMatricula(string);
    c. void setMatricula();
    d. string setMatricula(string);

2. La firma del constructor por omisión de la clase ***Examen*** sería:
    a. void Examen();
    b. void Examen(){};
    c. Examen();
    d. Examen Examen();

3. En el siguiente segmento de código en C++, completa lo que se te pide atendiendo la sintaxis correcta del lenguaje.

```
class MiClase {
    int var1;
    public:
        MiClase();
    _____ //Coloca aquí la declaración o firma del método setVar1
    _____ //Coloca aquí la declaración o firma del método getVar1
    …….
};
```

Diseña el contenido del constructor por omisión y de los métodos *setVar1* y *getVar1*. Inicializa a *var1* en 101;

```
// Coloca aquí el código que implementa al constructor por omisión.
// Coloca aquí el código que implementa al método setVar1
// Coloca aquí el código que implementa al método getVar1
```

4. Siguiendo con el ejemplo anterior, completa la siguiente aplicación que utiliza MiClase.

```
#include “Miclase.h”

// Declara la constante entera N con valor 10

int main()
{
        _____; // Declara el objeto miObjeto de tipo MiClase
        _____; // Cambia por 1500 el valor de var1 de miObjeto
        _____; // Muestra en pantalla el contenido de var1 de miObjeto
        _____; // Declara el arreglo miArreglo con dimensión N de tipo MiClase
        _____; // Muestra el contenido de var1 del elemento 5 de miArreglo
        return 0;
}
```

5. Usando el estándar revisado en clase, diseña el diagrama de clases de **MiClase**.

Práctico. 70 %

1. **Diseña** e **implementa** en **C++** la **clase** modelada en el diagrama de la figura 1. Revisa en el apartado Especificaciones algunos requerimientos básicos que deberás considerar.
2. **Diseña** los **casos** de **prueba** que te permitan probar casos generales de tu clase.
3. **Coloca** en el **diagrama** (figura 1) el tipo de relación adecuada entre las clases si en la clase Empleado se utiliza un atributo de clase Fecha.
4. En función de los casos de prueba, **agrega** una **aplicación** en consola que permita implementar los casos de prueba.

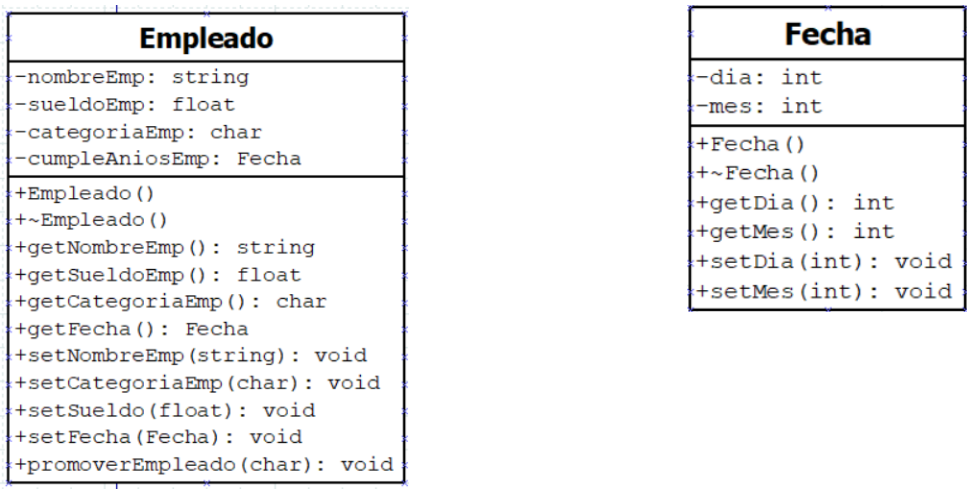

Figura 1. Diagrama de clases. Clase Empleado y clase Fecha.

Especificaciones.

1. Debes incluir los **constructores** señalados para cada clase.
2. Los únicos **valores permitidos** para el atributo *categoriaEmp* son A, B, C y D.
3. Debes incluir métodos ***modificadores*** y ***accesores*** requeridos para ambas clases.
4. ***promoverEmpleado***. El método recibe como parámetro una categoría válida. Si la categoría recibida es mayor a la que el empleado tenía, su sueldo debe incrementarse 2.5% por cada avance de categoría.

   Ejemplos:

   Si llegara un valor de categoría no válido, es decir fuera del rango A, B, C o D, el método debe mostrar un mensaje de error “Categoría inválida” y no modificar el sueldo del empleado.

   Si la categoría actual es A y el valor del parámetro es C, el porcentaje de incremento del sueldo sería 5 %.

   Si la categoría actual es C y el valor del parámetro es C, B o A, el sueldo NO se incrementa.

   Si la categoría actual es A y el valor del parámetro es D, el porcentaje de incremento del sueldo sería del 7.5 %.
5. En tu aplicación,
   a. Declara dos empleados, *emp1* y *emp2*. El empleado emp1 debe llamarse “Pepe”, ganar 10000, tener una laboral categoría ´A´ y su cumpleaños es mayo 19. El empleado emp2

debe llamarse “Rocio”, ganar 12000, tener una categoría laboral B y cumplir años el 1 de enero.

b. Muestra en pantalla los datos completos de los dos empleados. Acomoda los datos de cada empleado en un mismo renglón.
c. Promueve a Pepe a la categoría D.
d. Muestra en pantalla nombre, categoría y sueldo de Pepe.
e. Muestra en pantalla la fecha del cumpleaños de Rocio.