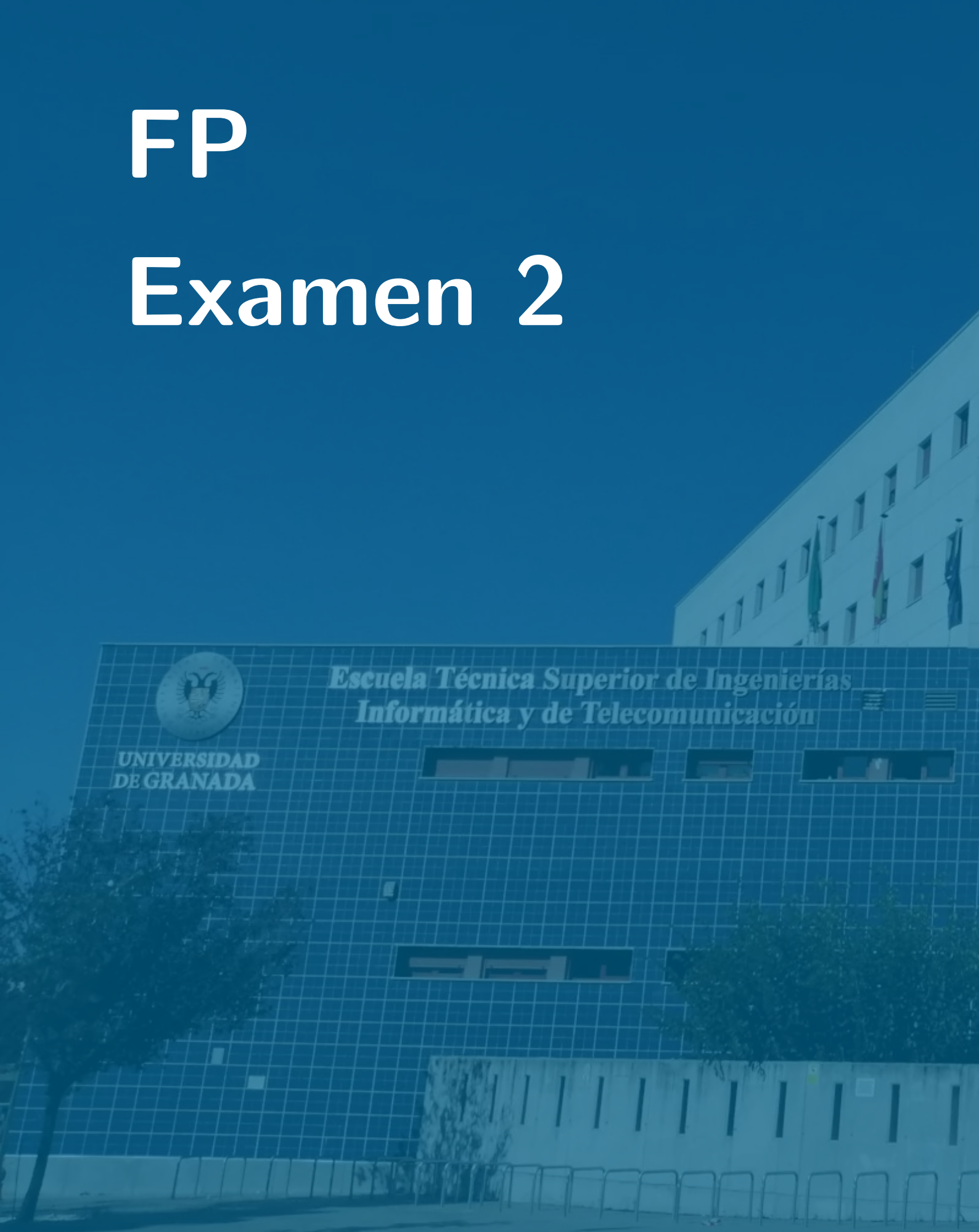
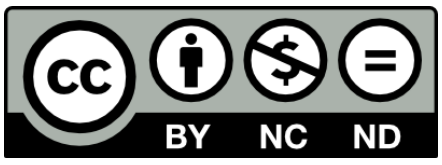
FP
Examen 2



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional (CC BY-NC-ND 4.0).

Eres libre de compartir y redistribuir el contenido de esta obra en cualquier medio o formato, siempre y cuando des el crédito adecuado a los autores originales y no persigas fines comerciales.

FP
Examen 2

Los Del DGIIM, `losdeldgiim.github.io`

Granada, 2026

Asignatura Fundamentos de Programación.

Curso Académico 2026/2027.

Grado Doble Grado en Ingeniería Informática y Matemáticas.

Grupo Único.

Profesor Francisco José Cortijo Bon.

Descripción Prueba práctica 1.

Fecha Noviembre de 2026.

Duración 75 minutos.

Normas para la realización del examen:

- Crear una carpeta en el escritorio llamada `EXAMEN` y copiar en ella el fichero `examen.cpp`. Modificar los datos que aparecen en la cabecera y escribir la solución en ese fichero.
- La entrega de la solución se hará durante el periodo de tiempo en el que se realiza el examen y se efectuará únicamente desde los ordenadores instalados en el aula a través de PRADO. Deberá entregar SU fichero `examen.cpp` (¡¡no se confunda!!).

Ejercicio 1 (10 puntos)**. Gestión de palabras en cadenas de caracteres**

Definición de palabra

Una palabra es una cadena de caracteres (un dato `string`) que **no** contiene caracteres separadores. Los separadores son los caracteres para los que la función `isspace` devuelve `true`, básicamente los caracteres espacio (`' '`), tabulador (`'\t'`) ó nueva línea (`'\n'`).

En esta prueba deberá escribir funciones relacionadas con la **gestión de palabras**.
Debe decidir la cabecera apropiada para cada una de ellas:

1. `EsPalabra`: Calcula (y devuelve) si un dato `string` es una **palabra**.
2. `CuentaPalabras`: Calcula (y devuelve) el número de **palabras** que forman un dato `string`.
3. `ObtenerPalabra`: Devuelve una **palabra** de un dato `string`. La función recibe el *número* de la palabra deseada teniendo en cuenta que 1 indica la primera palabra, 2 la segunda, etc.
4. `AMayuscula`: Transforma a mayúsculas todos los caracteres que conforman la **palabra** sobre la que trabaja la función y devuelve la cadena transformada.
5. `ComprimeCadena`: La compresión consiste en *eliminar los separadores excesivos de una cadena, o sea, todos los separadores iniciales y finales, así como todos los intermedios (aquellos que aparecen entre cada par de palabras dejando únicamente un espacio en blanco entre cada par de palabras.*

   La función devuelve la cadena resultante de comprimir la cadena sobre la que se trabaja.
6. `AMayusculaComprimeCadena`: Devuelve una nueva cadena, resultante de la compresión de la cadena sobre la que se trabaja con todos sus caracteres en mayúsculas.

Tareas a implementar

1. Implemente las funciones indicadas anteriormente.
2. Implemente las funciones auxiliares que estime oportuno.

Indicaciones y restricciones

1. Modularice el código utilizando **funciones**. Evite la repetición de código. Se valorará especialmente la calidad de la modularización en base a su facilidad de uso por otras aplicaciones y/o otras funciones.

2. Ninguna de las funciones indicadas modifica el dato `string` sobre el que trabaja.

3. Si no se indica otra cosa, no hay ninguna restricción sobre el dato `string` sobre el que se trabaja.

4. Especifique claramente las **precondiciones** de las funciones (si las hubiera). Se valorará su expresión, conveniencia y justificación. Comente correctamente funciones y código.

5. No puede utilizarse contenido no explicado en los temas 1, 2 ó 3. Les recuerdo la prohibición expresa de emplear cualquier elemento de programación no estructurada (`continue`, `exit`, `break`,...).

Observación. Se insta al lector a resolver este examen y a convertirlo a formato Markdown, para que pueda subirse a la web.