APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 23/10/19	DNI		

Duración: dos horas

Escribir únicamente con bolígrafo de tinta negra o azul, excepto en los casos donde se desee crear diagramas de flujo o pseudocódigo.

Todos los dispositivos electrónicos deben estar guardados en la mochila y apagados o sin sonido. En caso contrario debe indicarse al profesor los motivos por los que debe tenerse el teléfono con sonido antes de comenzar el examen.
El uso de cualquier dispositivo electrónico o el plagio o copia conllevará el suspenso del examen y de la evaluación ordinaria, acudiendo directamente a la evaluación extraordinaria según el artículo 9 del reglamento sobre pruebas de evaluación y su revisión.

Artículo 9.º. De la utilización de métodos ilícitos para la superación de las pruebas de evaluación.

"Cualquier evidencia de plagio, de copia del examen de un compañero, o cualquier intento de obtener de forma fraudulenta las respuestas a las preguntas de una prueba de evaluación, será penalizada y podrá suponer la apertura de expediente y la aplicación de las correspondientes sanciones, pudiendo llegar a sustanciarse en la expulsión de la Universidad del alumno o alumnos implicados, conforme a lo establecido en los arts. 29 y siguientes del Reglamento del Alumnado de esta Universidad. "

El diagrama de flujo y pseudocódigo puede apoyarse en comentarios utilizando la #, tal y como hace el lenguaje de programación Python.

Rellenar los datos en todas las páginas del examen.

Resultados de aprendizaje que se evalúan en este examen:

Entender los conocimientos básicos de algorítmica y complejidad computacional.
Ser capaz de realizar análisis de complejidad de algoritmos.
Ser capaz de implementar algoritmos básicos.
Ser capaz de implementar algoritmos utilizando técnicas de desarrollo de algoritmos.

Ejercicio 1 (2 puntos) Tiempo estimado: 20 minutos. Responda a las cuestiones

Respuestas correctas suman 0.5 puntos. Se debe responder a 4 preguntas de las 5 disponibles. Las 4 preguntas a contestar son elegidas libremente por el estudiante.

1) Calcula la complejidad computacional de las siguientes funciones y ordénalas según orden creciente:

   a) $4n * \log(n) + 2n$
   b) 2^4
   c) 2^n
   d) $3n + 100 \log(n)$
   e) $n^3 + n^2 + 10$
   f) $\log(n)$

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
			GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

2) Dadas dos funciones y su número de operaciones:

A =200n
B = $2n^2$

Calcular a partir de qué tamaño de entrada A es más eficiente que B.

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
			GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

3) Explica las diferencias entre la recursión lineal final y la recursión lineal no final y nombra un ejemplo de cada una.

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
			GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

4) Explica brevemente el método de la búsqueda binaria ¿Qué prerrequisitos debe cumplir?

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 23/10/19	DNI		

5) Dibuja las diferencias iteraciones del algoritmo de Quicksort con un ejemplo práctico donde se elegirá como pivote el primer elemento de la lista para el siguiente conjunto de entrada.

50 20 84 13 22 16 89 85

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA	Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI	**GIN 02**	

Ejercicio 2 (1 punto) Tiempo estimado: 10 minutos. Dado el siguiente algoritmo de búsqueda en Python (sequentialSearch):

```
1  def sequentialSearch(conjuntoBusqueda, datoABuscar):
2      ''' Function to search the datoAbuscar in the collection conjuntoBusqueda
3      :param conjuntoBusqueda: iterable collection to search the element datoAbuscar
4      :param datoABuscar: element to search
5      :type conjuntoBusqueda iterable collection of Integers, Floats or doubles
6      :type datoABuscar: Integer, Float or Double
7      :return the position (<lenSize) of target if target is found in indicated
       portion of a Python list.
8          Return lenSize if target is not found in the Python List
9      :rtype: Integer
10     '''
11     i=0
12     longitudSecuencia = len(conjuntoBusqueda)
13     while ((i < longitudSecuencia) and (conjuntoBusqueda[i] != datoABuscar)):
14       i = i+1
15     return i
```

Modificar el algoritmo para que realice una búsqueda más eficiente mediante la utilización de un centinela

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA	Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI	**GIN 02**	

Ejercicio 3 (1 punto) Tiempo estimado: 15 minutos. La suma de prefijos consiste en la suma de los j primeros números de una secuencia para todos los elementos de la secuencia. Esto es, para la secuencia S de números enteros o reales se devuelve una secuencia A (prefixSums in the code) de números enteros o naturales según el tipo de datos contenidos en S:

$$A[j] = \sum_{i=0}^{j} S[i]$$

Dado el siguiente algoritmo en Python que calcula la suma prefijos para todos los elementos de tipo int, float o double de una colección iterable.

```
1  def prefixSumsn2(S):
2      """
3      Function to calculate the prefixSums in the collection S.
4      Example:  Input  : arr[] = {10, 20, 10, 5, 15}
5                Output : prefixSum[] = {10, 30, 40, 45, 60}
6                prefixSum[0] = 10,
7                prefixSum[1] = prefixSum[0] + prefixSum[1] = 30,
8                prefixSum[2] = prefixSum[0] + prefixSum[1] + prefixSum[2] = 40 and so
                 on.
9      :param S iterable collection to search the element datoAbuscar
10     :return the prefixSums collection calculated by the addition of the
11     :rtype: Iterable Collection of Ints, Floats or Doubles, depending on S data type
12
13     """
14     n = len(S)
15     prefixsums = [0] * n
16     total = 0
17     for i in range(n):
18       total=0
19       for j in range(1+i):
20         total += S[j]
21       prefixsums[i] = total
22     return prefixsums
```

Pregunta:
Calcula su orden de complejidad computacional y su complejidad computacional en el caso-peor, el caso-mejor y el caso-medio. Explicar cuando se producirían dichos casos.

Respuesta:

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 23/10/19	DNI		

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
ASIGNATURA	Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI	**GIN 02**	

Ejercicio 4 (2 puntos) Tiempo estimado: 15 minutos. La suma de prefijos consiste en la suma de los j primeros números de una secuencia para todos los elementos de la secuencia. Esto es, para la secuencia S de números enteros o reales se devuelve una secuencia A (prefixSums in the code) de números enteros o naturales según el tipo de datos contenidos en S:

$$A[j] = \sum_{i=0}^{j} S[i]$$

Dado el siguiente algoritmo en Python que calcula la suma prefijos para todos los elementos de tipo int, float o double de una lista.

```
1  def prefixSumsn2(S):
2      """
3      Function to calculate the prefixSums in the collection S.
4      Example:  Input  : arr[] = {10, 20, 10, 5, 15}
5                Output : prefixSum[] = {10, 30, 40, 45, 60}
6                prefixSum[0] = 10,
7                prefixSum[1] = prefixSum[0] + prefixSum[1] = 30,
8                prefixSum[2] = prefixSum[0] + prefixSum[1] + prefixSum[2] = 40 and so
                 on.
9      :param S iterable collection to search the element datoAbuscar
10     :return the prefixSums collection calculated by the addition of the
11     :rtype: Iterable Collection of Ints, Floats or Doubles, depending on S data type
12
13     """
14     n = len(S)
15     prefixsums = [0] * n
16     total = 0
17     for i in range(n):
18       total=0
19       for j in range(1+i):
20         total += S[j]
21       prefixsums[i] = total
22     return prefixsums
```

Pregunta:
Diseña e implementa un algoritmo más eficiente para calcular la suma de prefijos. En este caso, la documentación y el control de excepciones no es obligatorio y no se va a valorar.

Respuesta:

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
			GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 23/10/19	DNI		

Ejercicio 5 (2 puntos) Tiempo estimado: 30 minutos. Suponed que sois la nueva incorporación de la tienda digital XXYYZZ para diseñar el flujo de negocio con su venta online. Dentro de la tienda digital, se han identificado algunas acciones que los clientes van a tener que realizar. Entre ellas la siguiente operación:

- Comprar (artículo, tarjeta, precio)

Esta operación de comprar debe verificar que el artículo tiene stock y que la tarjeta tiene crédito disponible sin haber llegado al límite.
Dibujad el diagrama de flujo (0.75 puntos) de la operación de comprar. Tras el diseño, donde se especificarán ciertas acciones, la implementación de pseudocódigo va a ser mucho más sencilla, y vamos a identificar nuestras funciones modulares. Por ello, procede a implementar el pseudocódigo o código Python de la operación de transferir, implementando todas las funciones modulares que forman parte de tu diseño (1.25 puntos). Toda implementación debe contener su documentación. Sin embargo, en este caso el control de excepciones se va a limitar a la documentación, donde especificaremos las excepciones que podrían ocurrir y en qué casos, sin ser necesaria su implementación.
Las funciones de acceso a los datos de la base de datos se suponen implementadas y se pueden utilizar con la sintaxis, por ejemplo, de GetNumeroCuentaAsociada(tarjeta), que devuelve el número de cuenta asociado a la tarjeta. Todas esas operaciones de obtener datos correspondientes a clientes o tarjetas se presuponen implementadas.

APELLIDOS	NOMBRE		GRUPO	CALIFICACIÓN
			GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA Algoritmos y Estructuras de Datos		FECHA 23/10/19	DNI		

Ejercicio 6 (2 puntos) Tiempo estimado: 30 minutos. Una lista reversa consiste en la ordenación de los elementos de la forma inversa. Diseña e implementa el pseudocódigo de una función recursiva que devuelva una nueva lista inversa a la lista introducida como parámetro S(n) u ordene la lista recibida como parámetro de forma inversa S(1).
La implementación del algoritmo con una nueva lista S(n) en lugar de la ordenación en la propia lista S(1) será puntuable hasta 1 punto.

APELLIDOS		NOMBRE		GRUPO	CALIFICACIÓN
				GIN 02	
ASIGNATURA	Algoritmos y Estructuras de Datos	FECHA 23/10/19	DNI		

Entregar esta hoja con el nombre y apellidos del alumno, DNI y grupo.