Examen final 2021-08-13

95.14/75.40 - Algoritmos y Programación I - Curso Essaya

Objetivo

Se dispone de los archivos `ej1.py`, `ej2.py`, `ej3.py`, `ej4.py` y `ej5.c` correspondientes a los 5 ejercicios del examen.

Cada uno tiene un lugar para escribir la implementación del ejercicio, y una función de pruebas para verificar que la solución es correcta.

El examen se aprueba con al menos 3 ejercicios correctamente resueltos. Un ejercicio se considera correctamente resuelto si:

- El programa `ej<n>` no tiene errores de sintaxis y puede ser ejecutado
- La implementación cumple con lo pedido en el enunciado

En algunos ejercicios se incluye un ejemplo de uno o dos casos de prueba y queda a cargo del alumno agregar más casos de prueba, para los que se provee sugerencias. En otros ejercicios se provee únicamente sugerencias. La implementación de las pruebas adicionales es **opcional**, pero se recomienda hacerlo ya que permite asegurar que la resolución del ejercicio es correcta.

Ejercicios en lenguaje Python

Al ejecutar cada uno de los ejercicios (`python3 ej<n>.py`), se ejecutan todas las pruebas presentes en la función `pruebas`.

Si alguna de las verificaciones falla, se imprime un mensaje de error y el programa termina su ejecución. Por ejemplo:

```
$ python3 ej1.py
Traceback (most recent call last):
File "ej1.py", line 148, in pruebas
    assert p != None
AssertionError
```

Cuando todas las pruebas pasan correctamente, se imprime `OK`:

```
$ python3 ej1.py
ej1.py: OK
```

Pruebas

Se recomienda usar la instrucción `assert` de la biblioteca estándar para verificar condiciones en las pruebas. Ejemplo de uso:

```python
# función a probar
def sumar(a, b):
    return a + b

# pruebas
```

```python
def pruebas():
    assert sumar(0, 0) == 0
    assert sumar(2, 3) == 5
    assert sumar(2, -2) == 0

    from os import path
    print(f"{path.basename(__file__)}: OK")

pruebas()
```

Nota: A veces para depurar un error en las pruebas es útil imprimir valores; se permite el uso de `print()` para ello.

Nota: A veces para implementar las pruebas es útil utilizar números aleatorios. Se permite el uso de la biblioteca `random` para ello. En ese caso, se recomienda ejecutar `random.seed(0)` al inicio del programa para asegurar que la secuencia de números aleatorios sea siempre la misma, y así facilitar la depuración.

Ejercicios en lenguaje C

Para compilar y ejecutar el ejercicio `ej5.c`:

```
$ gcc -Wall -pedantic -std=c99 ej5.c -o ej5
$ ./ej5
ej5.c: OK
```

Pruebas

Se recomienda usar la función `assert` de la biblioteca estándar para verificar condiciones en las pruebas. Ejemplo de uso:

```c
#include <stdio.h>
#include <assert.h>

// funcion a probar
int sumar(int a, int b) {
    return a + b;
}

// pruebas
int main(void) {
    assert(sumar(0, 0) == 0);
    assert(sumar(2, 3) == 5);
    assert(sumar(2, -2) == 0);

    printf("%s: OK\n", __FILE__);
    return 0;
}
```

Ejercicios

Ejercicio 1 Implementar en forma **recursiva** la función `merge(a, b)`, que recibe dos listas ordenadas y devuelve una lista con los elementos intercalados ordenadamente, en **tiempo lineal**.

Ejercicio 2

a. Implementar una función `generar(n)` que genera una lista de N números aleatorios, usando para ello cualquiera de las funciones del módulo `random`.

b. Implementar la función `ordenar(a)`, que ordena la lista `a` utilizando el algoritmo según el siguiente pseudocódigo:

```
algoritmo ordenar(arreglo A con N elementos):
    repetir N veces:
        para i := 1 hasta N-1 inclusive:
            si A[i-1] > A[i] entonces:
                intercambiar(A[i-1], A[i])
```

Indicar (en un comentario) cuál es la complejidad computacional en tiempo (`T(N)`) y espacio (`E(N)`) del algoritmo.

Ejercicio 3 Escribir una función que recibe la ruta de un archivo CSV de la forma `comuna,DNI` (no ordenado bajo ningún criterio particular), y por cada comuna existente en el archivo escribe un archivo CSV con nombre `<comuna>.csv` conteniendo el listado de DNIs de personas de esa comuna.

Nota: **No** se puede asumir que el contenido completo del archivo entra en la memoria de la computadora.

Ejercicio 4 Escribir el método de `ListaEnlazada` `lshift(n)` que rota la lista `n` posiciones a la izquierda, siendo $0 \leq n \leq len(L)$. Ejemplo:

```
L = ListaEnlazada([1, 2, 3, 4, 5, 6])
L.lshift(2)
# L ahora es [3, 4, 5, 6, 1, 2]
```

Ejercicio 5 Escribir en lenguaje C la función `void title_case(char s[])` que convierte la cadena a un título. En un título, todas las palabras comienzan con una mayúscula y siguen con minúsculas.

Ejemplo: `title_case("HoLa QuE tAl") -> "Hola Que Tal"`

Utilizar la función `isalpha` para determinar si un caracter es parte de una palabra o no, y las funciones `toupper` y `tolower` para convertir a mayúscula y minúscula. Estas funciones están en el módulo `ctype.h`.