Banco enunciados ALGORITMOS:

TEMA 3:

Ejercicio 1

En el momento de la creación del mundo, los sacerdotes del templo de Brahma recibieron una plataforma de bronce sobre la cual había tres agujas de diamante. En la primera aguja estaban apilados setenta y cuatro discos de oro, cada una ligeramente menor que la que estaba debajo. A los sacerdotes se les encomendó la tarea de pasarlos todos desde la primera aguja a la tercera, con dos condiciones, solo puede moverse un disco a la vez, y ningún disco podrá ponerse encima de otro más pequeño. Se dijo a los sacerdotes que, cuando hubieran terminado de mover los discos, llegaría el fin del mundo. Resolver este problema de la Torre de Hanói.

Ejercicio 2

Realiza el código para calcular el determinante de una matriz cuadrada de [3 x 3], regla de Sarrus de forma recursiva y de forma iterativa

Ejercicio 3

Dada una lista de las naves (y vehículos) de Star Wars –consideraremos a todos como naves– de las que conocemos su nombre, largo, tripulación y cantidad de pasajeros, desarrollar los algoritmos necesarios para resolver las actividades detalladas a continuación: realizar un listado ordenado por nombre de las naves de manera ascendente y por largo de las mismas de manera descendente; • mostrar toda la información del “Halcón Milenario” y la “Estrella de la Muerte”; • determinar cuáles son las cinco naves con mayor cantidad de pasajeros; • indicar cuál es la nave que requiere mayor cantidad de tripulación; • mostrar todas las naves que comienzan con AT; • listar todas las naves que pueden llevar seis o más pasajeros; • mostrar toda la información de la nave más pequeña y la más grande.

Ejercicio 4

Implementar sobre el TDA polinomio desarrollado previamente las siguientes actividades: • restar dos polinomios; • dividir el polinomio por un numero; • eliminar un término; • determinar si un término existe en un polinomio.

Tema 4

Ejercicio 1

Desarrollar los algoritmos necesarios para generar un árbol de Huffman a partir de la siguiente tabla –para lo cual deberá calcular primero las frecuencias de cada carácter a partir de la cantidad de apariciones del mismo–, para resolver las siguientes actividades: a. la generación del árbol debe hacerse desde los caracteres de menor frecuencia hasta los de mayor, en el caso de que dos caracteres tengan la misma frecuencia, primero se toma el que este primero en el alfabeto, el carácter “espacio” y “coma” se consideraran anteúltimo y último respectivamente en el orden alfabético; b. descomprimir los siguientes mensajes –cuyo árbol ha sido construido de la misma manera que el ejemplo visto anteriormente:

I. Mensaje 1:
“10001011101011000010111010001110000011011000000111100111101001011000011010011100110100010111010111111101000011110011111100111101000110000110000
00101101011110111111101110101101101110011101101111001111111001010010100101000001
01101011000101100110100011100100101100001100100011010110101011111111110110111
01110010000100101011000111111100010001110110011001011010001101111101011010001101
11011100000000111001001010100011111100001100101101011100110011110100011000110 00000101101011111001110011100”

II. Mensaje 2:
“01101010110111001010001111010111001101110101101101000010001110101001011110100111111101110010100011110101110011011101011000011000100110100011100100 100011000101100110011100100100001111011111010” c. finalmente, calcule el espacio de memoria requerido por el mensaje original y el comprimido. Carácter Cantidad Frecuencia A 11 B 2 C 4 D 3 E 14 G 3 I 6 L 6 M 3 N 6 O 7 P 4 Q 1 R 10 S 4 T 3 U 4 V 2 ‘ ’ espacio 17 , coma 2

Ejercicio 2

El comandante de la estrella de la muerte el gran Moff Tarkin debe administrar las asignaciones de vehículos y Stromtroopers a las distintas misiones que parten desde la estrella de la muerte, para facilitar esta tarea nos encomienda desarrollar las funciones necesarias para gestionar esto mediante prioridades de la siguiente manera: a. de cada misión se conoce su tipo (exploración, contención o ataque), planeta destino y general

que la solicitó; b. si la misión fue pedida por Palpatine o Darth Vader estas tendrán alta prioridad, sino su prioridad será baja; c. si la misión es de prioridad alta los recursos se asignarán manualmente independiente- mente de su tipo; d. si la misión es de baja prioridad se asignarán los recursos de la siguiente manera depen- diendo de su tipo:

- 1. exploración: 15 Scout Troopers y 2 speeder bike,
- 2. contención: 30 Stormtroopers y tres vehículos aleatorios (AT-AT, AT-RT, AT-TE, AT-DP, AT-ST) pueden ser repetidos,
- 3. ataque: 50 Stormtroopers y siete vehículos aleatorios (a los anteriores se le suman AT-M6, AT-MP, AT-DT), e. realizar la atención de todas las misiones y mostrar los recursos asignados a cada una, permitiendo agregar nuevos pedidos de misiones durante la atención; f. indicar la cantidad total de recursos asignados a las misiones.

Ejercicio 3

Se requiere implementar una red de ferrocarriles compuesta de estaciones de trenes y cambios de agujas (o desvíos). Contemplar las siguientes consideraciones: a. cada vértice del grafo no dirigido tendrá un tipo (estación o desvió) y su nombre, en el caso de los desvíos el nombre es un número –estos estarán numerados de manera consecutiva–; b. cada desvío puede tener múltiples puntos de entrada y salida; c. se deben cargar seis estaciones de trenes y doce cambios de agujas; d. cada cambio de aguja debe tener al menos cuatro salida o vértices adyacentes; e. y cada estación como máximo dos salidas o llegadas y no puede haber dos estaciones conectadas directamente; f. encontrar el camino más corto desde:

- 1. la estación King's Cross hasta la estación Waterloo,
  2. la estación Victoria Train Station hasta la estación Liverpool Street Station,
- 3. la estación St. Pancras hasta la estación King's Cross; Ejercicio
- 4. Desarrollar un algoritmo numérico iterativo que permita calcular el método de la bisección de una función f(x). Desarrollar un algoritmo numérico iterativo que permita calcular el método de la secante de una función f(x). Desarrollar un algoritmo numérico iterativo que permita calcular el método de Newton-Raphson de una función f(x). Comparar los tres algoritmos anteriores para resolver la siguiente función: x3 + x +16 = 0, respecto de la cantidad de iteraciones necesarias por cada método para converger. ¿Cuánto es la diferencia en decimales entre las distintas soluciones? Método Numero de iteraciones Solución Secante Bisección Newton-Raphson

Tema 5

Ejercicio 1

Crea el siguiente módulo:

- El módulo se denominará operaciones.py y contendrá 4 funciones para realizar una suma, una resta, un producto y una division entres dos números. Todas ellas devolverán el resultado.

- En las funciones del módulo deberá de haber tratamiento e invocación manual de errores para evitar que se quede bloqueada una funcionalidad, eso incluye:

    - TypeError: En caso de que se envíen valores a las funciones que no sean números. Además, deberá aparecer un mensaje que informe Error: Tipo de dato no válido.

    - ZeroDivisionError: En caso de realizar una división por cero. Además, deberá aparecer un mensaje que informe Error: No es posible dividir entre cero.

Una vez creado el módulo, crea un script calculos.py en el mismo directorio en el que deberás importar el módulo y realizar las siguientes instrucciones. Observa si el comportamiento es el esperado:

```
from operaciones import *

a, b, c, d = (10, 5, 0, "Hola")

print( "{} + {} = {}".format(a, b, suma(a, b) ) )
print( "{} - {} = {}".format(b, d, resta(b, d) ) )
print( "{} * {} = {}".format(b, b, producto(b, b) ) )
print( "{} / {} = {}".format(a, c, division(a, c) ) )
```

Ejercicio 2

¿Eres capaz de desarrollar un reloj de horas, minutos y segundos utilizando el módulo datetime con la hora actual? Hazlo en un script llamado reloj.py y ejecútalo en la terminal:

Ayuda

El módulo time integra una función llamada sleep(segundos) que puede pausar la ejecución de un programa durante un tiempo. Así mismo el módulo os integra la función system('cls') y system('clear') para limpiar la pantalla de la terminal en sistemas Windows y Linux/Unix respectivamente.

Ejercicio 3

Crea un script llamado generador.py que cumpla las siguientes necesidades:

- Debe incluir una función llamada leer_numero(). Esta función tomará 3 valores: ini, fin y mensaje. El objetivo es leer por pantalla un número >= que ini y <= que fin. Además a la hora de hacer la lectura se mostrará en el input el mensaje enviado a la función. Finalmente se devolverá el valor. Esta función

tiene que devolver un número, y tiene que repetirse hasta que el usuario no lo escriba bien (lo que incluye cualquier valor que no sea un número del ini al fin).

- Una vez la tengas creada, deberás crear una nueva función llamada generador, no recibirá ningún parámetro. Dentro leerás dos números con la función leer_numero():

- El primer numero será llamado numeros, deberá ser entre 1 y 20, ambos incluidos, y se mostrará el mensaje ¿Cuantos números quieres generar? [1-20]:

- El segundo número será llamado modo y requerirá un número entre 1 y 3, ambos incluidos. El mensaje que mostrará será: ¿Cómo quieres redondear los números? [1]Al alza [2]A la baja [3]Normal:.

- Una vez sepas los números a generar y cómo redondearlos, tendrás que realizar lo siguiente:

- Generarás una lista de números aleatorios decimales entre 0 y 100 con tantos números como el usuario haya indicado.

- A cada uno de esos números deberás redondearlos en función de lo que el usuario ha especificado en el modo.

- Para cada número muestra durante el redondeo, el número normal y después del redondeo.

- Finalmente devolverás la lista de números redondeados. El objetivo de este ejercicio es practicar la reutilización de código y los módulos random y math. Recordatorio El redondeo tradicional round() no requiere importar ningún módulo, es una función por defecto.

Ejercicios « Manejo de ficheros>>

Ejercicio 1

En este ejercicio deberás crear un script llamado personas.py que lea los datos de un fichero de texto, que transforme cada fila en un diccionario y lo añada a una lista llamada personas. Luego recorre las personas de la lista y para cada una muestra de forma amigable todos sus campos.

El fichero de texto se denominará personas.txt y tendrá el siguiente contenido en texto plano (créalo previamente):

```
1;Carlos;Pérez;05/01/1989
2;Manuel;Heredia;26/12/1973
3;Rosa;Campos;12/06/1961
4;David;García;25/07/2006
```

Los campos del diccionario serán por orden: id, nombre, apellido y nacimiento.

Importante Si quieres leer un fichero que no se ha escrito directamente con Python, entonces es posible que encuentres errores de codificación al mostrar algunos caracteres. Asegúrate de indicar la codificación del fichero manualmente durante la apertura como argumento en el open, por ejemplo, con UTF-8:

open(..., encoding="utf8")

Ejercicio 2

En este ejercicio deberás crear un script llamado contador.py que realice varias tareas sobre un fichero llamado contador.txt que almacenará un contador de visitas (será un número):

- Nuestro script trabajará sobre el fichero contador.txt. Si el fichero no existe o está vacío lo crearemos con el número 0. Si existe simplemente leeremos el valor del contador.
- Luego a partir de un argumento:
    - Si se envía el argumento inc, se incrementará el contador en uno y se mostrará por pantalla.
    - Si se envía el argumento dec, se decrementará el contador en uno y se mostrará por pantalla.
    - Si no se envía ningún argumento (o algo que no sea inc o dec), se mostrará el valor del contador por pantalla.
- Finalmente guardará de nuevo el valor del contador de nuevo en el fichero.
- Utiliza excepciones si crees que es necesario, puedes mostrar el mensaje: **Error: Fichero corrupto.**

Ejercicio 3

Utilizando como base el ejercicio de los personajes que hicimos, en este ejercicio tendrás que crear un gestor de personajes gestor.py para añadir y borrar la información de los siguientes personajes:

	Vida	Ataque	Defensa	Alcance
Caballero	4	2	4	2
Guerrero	2	4	2	4
Arquero	2	4	1	8

Deberás hacer uso del módulo pickle y todos los cambios que realices se irán guardando en un fichero binario personajes.pckl, por lo que aunque cerremos el programa los datos persistirán.

- Crea dos clases, una Personaje y otra Gestor.
- La clase Personaje deberá permitir crear un personaje con el nombre (que será la clase), y sus propiedades de vida, ataque, defensa y alcance (que deben ser números enteros mayor que cero obligatoriamente).
- Por otro lado la clase Gestor deberá incorporar todos los métodos necesarios para añadir personajes, mostrarlos y borrarlos (a partir de su nombre por ejemplo, tendrás que pensar una forma de hacerlo), además de los métodos esenciales para guardar los cambios en el fichero personajes.pckl que se deberían ejecutar automáticamente.
- En caso de que el personaje ya exista en el Gestor entonces no se creará.

Una vez lo tengas listo realiza las siguientes pruebas en tu código:

- Crea los tres personajes de la tabla anterior y añádelos al Gestor.
- Muestra los personajes del Gestor.
- Borra al Arquero.
- Muestra de nuevo el Gestor.

Sugerencia El ejemplo con pickle que realizamos anteriormente puede servirte como base