Examen 2023-01-17

Examen teórico

1 Ejercicio 1

El alfabeto Morse fue desarrollado por Alfred Vail y Samuel Morse en 1830 con la invención del telégrafo eléctrico. En esencia, es un sistema de representación de letras y números mediante códigos consistentes en rayas y puntos (es decir, señales telegráficas intermitentes que se diferencian en el tiempo de duración de la señal: larga o corta).

Figura 1: Tabla de codificación Morse

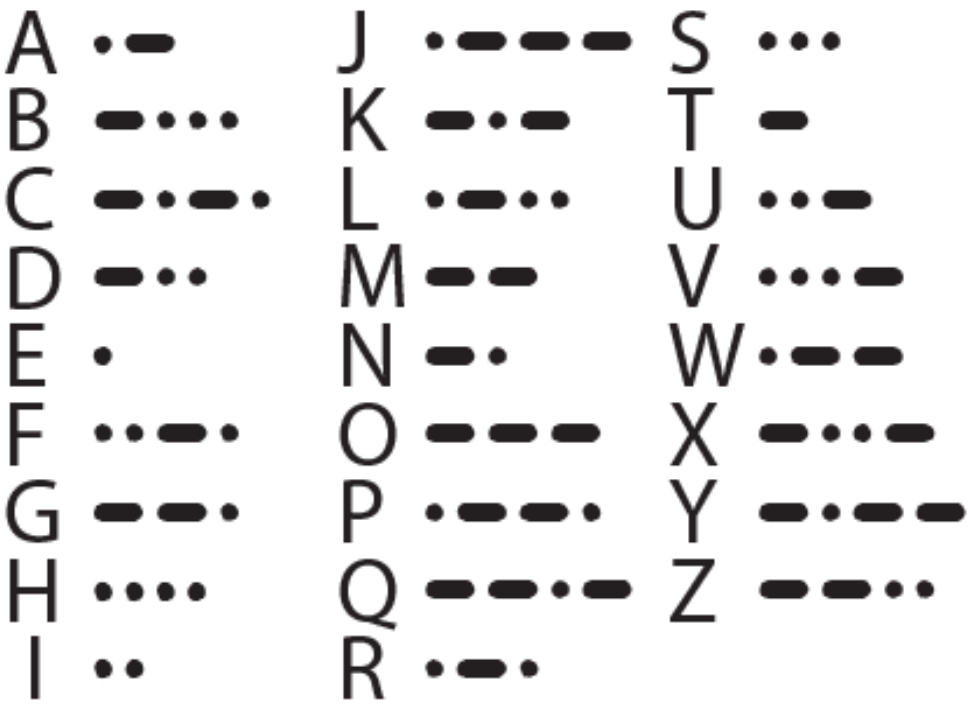

Figura 2: Telégrafo y tabla de codificación Morse de las letras mayúsculas.

Para este ejercicio deberás crear un programa que reciba un fichero de texto con la información codificada en Morse y muestre por pantalla el mensaje decodificado. Para simplificar, se asumirá que el mensaje decodificado sólo contendrá letras mayúsculas.

Para poder decodificar el mensaje, se dispone de un fichero secuencial de registros de tipo `Corresp`, llamado `morse.txt`, en el que cada linea contiene la codificación Morse de una letra mayúscula.

```
A .-
B -...
C -.-.
D -..
E .
F ..-.
G --.
H ....
I ..
J .---
K -.-
L .-..
M --
N -.
O ---
P .--.
Q --.-
R .-.
S ...
```

```
T -
U ..-
V ...-
W .--
X -..-
Y -.--
Z --..
```

En Python:

```python
import dataclasses

@dataclasses.dataclass
class Corresp:
    letra: str  # Un solo carácter
    codigo: str # Max 5 letras
```

- Desarrollar una función `cargar_codigo()` que lea del fichero `morse.txt` los códigos Morse correspondientes a cada letra, los guarde en la estructura de datos adecuada y la devuelva.
  - Se asumirá que el fichero es correcto y completo
  - No hay letras repetidas.
  - Las letras no están ordenadas.
- Desarrollar un programa que pida y lea de teclado el nombre de un fichero de texto que contiene un mensaje codificado en Morse, y muestre por pantalla el resultado de la decodificación. Notas:
  - En el fichero de texto, cada letra codificada está separada de la siguiente por el carácter `'/'`.
  - El fichero de texto puede tener varias líneas, en cuyo caso se mostrará el mensaje codificado también separado en esas líneas.
  - Para simplificar, en el mensaje codificado no habrá ningún carácter que no sean letras mayúsculas (no habrá espacios en blanco, ni signos de puntuación), y por tanto tampoco tiene que haberlos en el mensaje decodificado.

Ejemplo de fichero de texto `mensaje.txt`, escrito en código Morse

```
..../---/.../.-/--/../-./-../---/
./.../-/---/./.../../-./.-/.--./.--./../-.../.-/
..../.-/.../-/.-/.--./../-./--./---/
```

Ejemplo de ejecución

Introduce el nombre del fichero: **mensaje.txt**
Mensaje:
HOLAMUNDO
ESTOESUNAPRUEBA
HASTALUEGO

Solución

```
import dataclasses


@dataclasses.dataclass
class Corresp:
    codigo: str  # Maximo 5 chars
    letra: str  # 1 char mayuscula


def cargar_codigo():
    tabla = []
    f = open("morse.txt", "r")
    linea = f.readline()
    while linea != "":
        datos = linea.strip().split(" ")
        letra = datos[0]
        codigo = datos[1]
        tabla.append(Corresp(codigo, letra))
        linea = f.readline()
    f.close()
    return tabla


def buscar_letra(tabla, codigo):
    resultado = ""
    i = 0
    while i < len(tabla) and resultado == "":
        if tabla[i].codigo == codigo:
            resultado = tabla[i].letra
        i += 1
    return resultado


def decodificar_mensaje(fichero, tabla):
    resultado = ""
    f = open(fichero, "r")
    linea = f.readline()
    while linea != "":
        palabras = linea.strip().split("/")

        for i in range(len(palabras)):
            letra = buscar_letra(tabla, palabras[i])
            resultado += letra
        resultado += "\n"
        linea = f.readline()
    f.close()
    return resultado


print("Introduce el nombre del fichero: ")
nombre_fichero = input()
```

```
tabla = cargar_codigo()

print("Mensaje:")
print(decodificar_mensaje(nombre_fichero, tabla))
```

2 Ejercicio 3

Dada una matriz cuadrada de enteros de tamaño $n \times n$, se dice que es un **cuadrado mágico** si contiene todos los números enteros del 1 al n^2, sin repeticiones, y el resultado de sumar cada fila, el de sumar cada columna, y el de sumar las diagonales principales, es siempre el mismo (ver ejemplo de la figura, en el que la matriz es un cuadrado mágico y dicho resultado es 15).

El valor de dicho resultado para un cuadrado mágico se denomina *constante mágica* (C). Puede deducirse matemáticamente a partir del tamaño n, y viene dado por la siguiente fórmula:

$$C = \frac{n(n^2+1)}{2}$$

Se pide, usando el lenguaje Python:

- Implementar la función `es_cuadrado_magico(matriz)` que devuelva un valor booleano indicando si el dato pasado como parámetro es un cuadrado mágico o no.
  - Para serlo, debe cumplir las siguientes condiciones:
    - La matriz es cuadrada.
    - Las dos diagonales principales suman lo mismo (C).
    - Todas las filas suman lo mismo (C).
    - Todas las columnas suman lo mismo (C).
  - Puedes asumir que:
    - La variable `matriz` tiene el mismo número de celdas en cada vila, pero no que es cuadrada.

Ejemplo de uso

```
print(es_cuadrado_magico([[1, 2, 3], [4, 5, 6], [7, 8, 9]]))
print(es_cuadrado_magico([[2, 7, 6], [9, 5, 1], [4, 3, 8]]))
```

False
True

Se valorará el uso adecuado de procedimientos y funciones en la solución, así como que no exista código repetido innecesariamente. Se asume que la matriz es cuadrada, no hay que comprobar nada al efecto.

Solución

```
def suma_fila(matriz, fila):
    suma = 0
    for col in range(len(matriz)):
        suma = suma + matriz[fila][col]
```

```
    return suma

def suma_columna(matriz, col):
    suma = 0
    for fila in range(len(matriz)):
        suma = suma + matriz[fila][col]
    return suma

def suma_diagonal_principal(matriz):
    suma = 0
    for i in range(len(matriz)):
        suma = suma + matriz[i][i]
    return suma

def suma_diagonal_secundaria(matriz):
    suma = 0
    j = len(matriz) - 1  # Empezamos por la ultima
    for i in range(len(matriz)):
        suma = suma + matriz[i][j]
        j -= 1
    return suma

def es_cuadrado_magico(mat):
    n = len(mat)
    c = (n * (n**2 + 1)) // 2

    resultado = True

    if len(mat) != len(mat[0]):
        resultado = False
    else:
        # Comprobar filas
        for fila in range(n):
            if suma_fila(mat, fila) != c:
                resultado = False

        # Comprobar columnas
        for col in range(n):
            if suma_columna(mat, col) != c:
                resultado = False

        # Comprobar diagonales
        if suma_diagonal_principal(mat) != c:
            resultado = False
        if suma_diagonal_secundaria(mat) != c:
            resultado = False

    return resultado
```