NOMBRE: ALVARO BLANCO
LEGAJO: 10622
TRABAJO PRACTICO Pattern Matching

Ejercicio 1

```
def existChar(string, c):
  for char in string:
    if char == c:
      return True
  return False
```

Ejercicio 2

```
def isPalindrome(string):
  string = string.lower()
  return string == string[::-1]
```

Ejercicio 3

```
def mostRepeatedChar(string):
  char_count = {}

  for char in string:
    if char in char_count:
      char_count[char] += 1
    else:
      char_count[char] = 1

  max_count = 0
  most_repeated_char = None
  for char, count in char_count.items():
    if count > max_count:
      max_count = count
      most_repeated_char = char

  return most_repeated_char
```

Ejercicio 6

```
def verifyBalancedParentheses(s):
  stack = []
  for char in s:
    if char == '(':
      stack.append(char)
    elif char == ')':
      if len(stack) == 0 or stack[-1] != '(':
        return False
      stack.pop()
  return len(stack) == 0
```

Ejercicio 7

```
def reduceLen(string):
  stack = []

  for char in string:
    if stack and stack[-1] == char:
      stack.pop()
    else:
      stack.append(char)
  return ''.join(stack)
```

Ejercicio 8

```
def isContained(s, word):
  s_index = 0
  word_index = 0

  while s_index < len(s) and word_index < len(word):
    if s[s_index] == word[word_index]:
      word_index += 1
    s_index += 1

  return word_index == len(word)
```

Ejercicio 10

Paso 1: Necesitaremos 6 estados para representar todas las posibles combinaciones de coincidencias y no coincidencias del patrón P = "aabab". Los estados se nombran como: q0, q1, q2, q3, q4 y q5.

Paso 2: Definición de las transiciones: las transiciones entre los estados basándonos en el patrón P = "aabab".

- Desde el estado q0, si la entrada es 'a', iremos al estado q1. De lo contrario, permaneceremos en el estado q0.
- Desde el estado q1, si la entrada es 'a', iremos al estado q2. De lo contrario, volveremos al estado q0.
- Desde el estado q2, si la entrada es 'b', iremos al estado q3. De lo contrario, volveremos al estado q0.
- Desde el estado q3, si la entrada es 'a', iremos al estado q4. De lo contrario, volveremos al estado q0.
- Desde el estado q4, si la entrada es 'b', iremos al estado q5. De lo contrario, volveremos al estado q0.
- Desde el estado q5, independientemente de la entrada, permaneceremos en el estado q5.

◦ Paso 3: Definición del estado inicial y los estados de aceptación: En este caso, el estado inicial será q0 y el único estado de aceptación será q5. El estado q5 representa que hemos encontrado una coincidencia completa del patrón P = "aabab" en la cadena de texto T.

2. Demostración del funcionamiento del AEF en la cadena de texto T = "aaababaabaababaab":
   ◦ Paso 1: Iniciamos en el estado q0, que es el estado inicial.
   ◦ Paso 2: Leemos el primer carácter de T, que es 'a'. Como el estado actual es q0 y la entrada es 'a', según las transiciones definidas anteriormente, pasamos al estado q1.
   ◦ Paso 3: Leemos el siguiente carácter de T, que también es 'a'. El estado actual es q1 y la entrada es 'a', por lo que pasamos al estado q2.

◦ Paso 4: Leemos el siguiente carácter de T, que es 'a'. El estado actual es q2, pero la entrada no coincide con la transición definida desde q2. Por lo tanto, volvemos al estado q0.

◦ Paso 5: Leemos el siguiente carácter de T, que es 'b'. El estado actual es q0 y la entrada es 'b', por lo que permanecemos en el estado q0.

◦ Paso 6: Leemos el siguiente carácter de T, que es 'a'. El estado actual es q0 y la entrada es 'a', por lo que pasamos al estado q1.

◦ Paso 7: Repetimos los pasos 3 a 6 hasta llegar al final de la cadena de texto T.

Siguiendo los pasos anteriores, continuamos leyendo los caracteres de T y siguiendo las transiciones correspondientes en el AEF hasta que lleguemos al estado q5, que es el estado de aceptación. Si en algún momento la entrada no coincide con la transición desde el estado actual, volvemos al estado q0.

Ejercicio 12

```
def findPrefix(T, P):
  i = 0
  j = 0

  while i < len(T) and j < len(P):
    if T[i] == P[j]:
      i += 1
      j += 1
    else:
      break

  return T[:i]
```

Ejercicio 14

```
def kmp_search(text, pattern):
  n = len(text)
  m = len(pattern)
  f = [0] * m  # Inicializamos la tabla de fallos

  # Construimos la tabla de fallos
  j = 0
  for i in range(1, m):
    while j > 0 and pattern[i] != pattern[j]:
      j = f[j - 1]
    if pattern[i] == pattern[j]:
      j += 1
    f[i] = j

  # Realizamos la búsqueda del patrón en el texto
  i = 0
  j = 0

  while i < n:
    while i < n and j < m:
      if text[i] == pattern[j]:
        i += 1
        j += 1
      elif j > 0:
        j = f[j - 1]
      else:
        i += 1
```

```
    if j == m:
      return i - j  # Devolvemos el índice de la primera ocurrencia
    else:
      return -1  # No se encontró ninguna ocurrencia del patrón

def KMP(s1, s2):
  i = kmp_search(s1, s2)
  if i == -1:
    return None
  end = i + len(s2)
  occurrences = []

  for j in range(i, end):
    occurrences.append(j)

  return occurrences
```