Pattern Matching

Nombre: Enzo Palau

PARTE 1:

Ejercicio 7:

```python
"""Ejercicio 7"""

def reduceLen(String):
    #Reduce la longitud de una cadena removiendo iterativamente pares de caracteres repetidos.
    i=0
    while i<len(String) :
        if String[i+1]==String[i]:
            String=String[0:i]+String[(i+2):]
        i=i+1
    return String
```

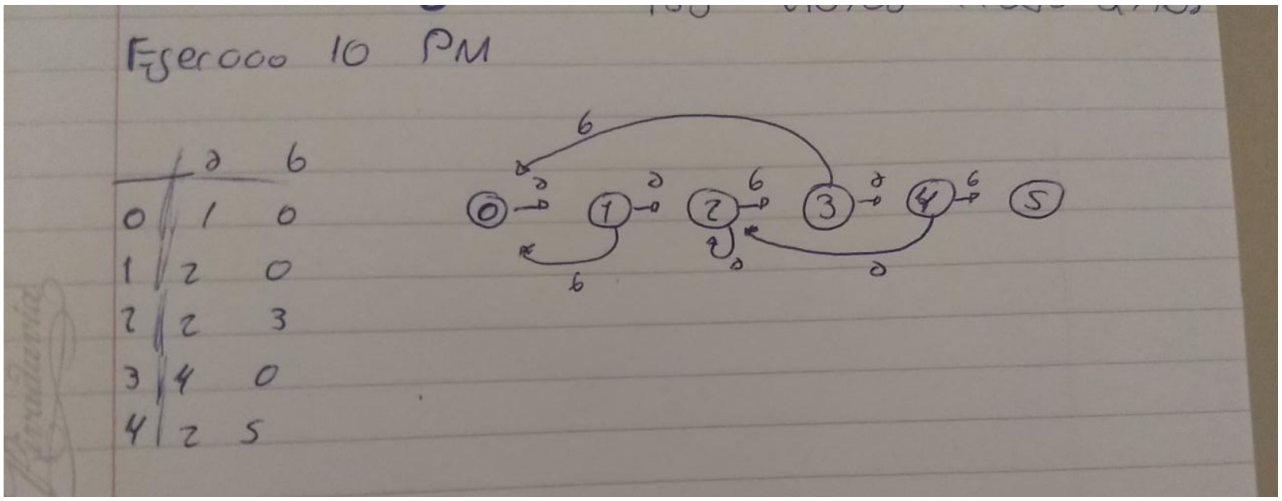
Ejercicio 8:

```python
"""Ejercicio 8"""

def isContained(String,St_pat):
    #Determina si los caracteres de una cadena se encuentran contenidos y en el mismo orden dentro de otra cadena
    cont=0
    for each in String:
        if cont==len(St_pat): return True
        if each is St_pat[cont]:
            cont+=1
    else: return False
```

PARTE 2:

Ejercicio 10:



Ejercicio 13:

```python
def Rhash(P):
    #calcula el hash con potencia de 128
    num=0
    Pow=len(P)-1
    for i in range(0,len(P)):
        num=num+(128**Pow)*ord(P[i])
        Pow=Pow-1
    return num


def rabin_karp(S,P):
    #algoritmo de Rabin-Karp
    m=len(P)
    n=len(S)
    hash_p=Rhash(P) #hash del patt
    for i in range(0,n-m+1): #O(n-m)
        t_s=S[i:i+m] #O(1)
        if Rhash(t_s)==hash_p and t_s==P:
            print(P,"Found at ",i)

```

Ejercicio 14:

```python
def KMP(T,P):
    #Implementa el algoritmo KMP
    pr=Compute_Prefix_Function(P)
    ocurrencias=[]
    q=0 #caracteres macheados
    for i in range(len(T)):
        while q>0 and P[q] != T[i]:
            q=pr[q-1] # no coincide entoces retroce
        if P[q]==T[i]:
            q=q+1 #el caracter coincide
        if q==len(P): #verifica que esta todo el patron visto
            ocurrencias.append(i-len(P)+1)
            print("Pattern occurs with shift",i-len(P)+1)
            q=pr[q-1] #seguimos viendo
    return ocurrencias

def Compute_Prefix_Function(P):
    #ve los mayores prefijos de p que son a la vez sufijos de Pq
    pi=[0]*(len(P))
    k=0
    for q in range(1,len(P)):
        while k>0 and P[k] != P[q]: #Este bucle busca el mayor prefijo válido anterior al sufijo actual P[q]
            k=pi[k]
        if P[k] is P[q]: #extendemos el prefijo valido
            k+=1
        pi[q]=k #almacena el valor de la función de prefijo calculado hasta ese punto
    return pi

```

Ejercicio 11:

En este ejercicio tendremos que simplemente copiar el código de el algoritmo de KMP como arriba pero con la condición de que en cada ciclo de del bucle for vemos si el valor de “q” es mayor que nuestro valor anterior, asi sacaremos el maximo