NAIVE-STRING-MATCHER(T, P)

```
1  n ← length[T]
2  m ← length[P]
3  for s ← 0 to n − m
4        do if P[1 . . m] = T[s + 1 . . s + m]
5              then print "Pattern occurs with shift" s
```

FINITT-AUTOMATION-MATCHER(T, δ, m)

```
1  n ← length[T]
2  q ← 0
3  for 1 ← 1 to n
4        do q ← δ(q, T[i])
5           if q = m
6              then print "Pattern occurs with shift" i − m
```

COMPUTE-TRANSITION-FUNCTION(P, Σ)

```
1  m ← length[P]
2  for q ← 0 to m
3        do for each charactor a ∈ Σ
4                 do k ← min (m + 1, q + 2)
5                    repeat k ← k − 1
6                       until P_k ⊐ P_q a
7                    δ(q, a) ← k
8  return δ
```

KMP-MATCHER(T,P)

```
1  n ← length[T]
2  m ← length[P]
3  Π ← COMPUTE-PREFIX-FUNCTION(P)
4  q ← 0                                   ▷ Number of characters matched
5  for i ← 1 to n                          ▷ Scan the next text from left to right
6        do while q > 0 and P[q + 1] ≠ T[i]
7              do q ← Π[q]                 ▷ Next character does not match
8           if P[q + 1] = T[i]
9              then q ← q + 1              ▷ Next Character matches
10          if q = m                    ▷ Is alll of P matched?
11             then print "Pattern occurs with shift" i − m
12                  q ← Π[q]               ▷ Look for next match
```

COMPUTE-PREFIX-FUNCTION(P)

```
1  m ← length[P]
2  Π[1] ← 0
3  k ← 0
4  for q ← 2 to m
5        do while k > 0 and P[k + 1] ≠ P[q]
6              do k ← Π[k]
7           if P[k + 1] = P[q]
8              then k ← k + 1
9           Π(q) ← k
10 return Π
```