Code Implementation

Program To Implement The KMP Algorithm In Python

In [8]:
```
import random
import string
import time
```

KMP Algorithm Implementation

In [21]:
```
def KMP(Pattern, Chars):
    start = time.time()
   # compute the start position (number of characters)of the longest suffix that matches the prefix
   # Then store prefix and the suffix into the list K, and then set the first element of K to be -1 and the second element to be 0
    K = [] # K[n] store the value so that if the mismatch happens at n, it should move pattern Pattern K[n] characters ahead.
    n = -1
    K.append(n) #add the first element, and keep n = 0.
    for k in range (1,len(Pattern) + 1):

        # traverse all the elements in Pattern, calculate the corresponding value for each element.
        while(n >= 0 and Pattern[n] != Pattern[k - 1]): # if n = 1, if n >=1 and the current suffix does not match then try a shorter suffix
            n = K[n]
        n = n + 1 # if it matches, then the matching position should be one character ahead
        K.append(n) #record the matching position for k

    #match the string Chars with Pattern
    m = 0
    for i in range(0, len(Chars)): #traverse through the list one by one
        while(m >= 0 and Pattern[m] != Chars[i]): # if they do not match then move Pattern forward with K[m] characters and restart the comparison
            m = K[m]
        m = m + 1 #if position m matches, then move forward with the next position
        if m == len(Pattern): # if m is already the end of K (or Pattern), then a fully matched pattern is found. Continue the comparison by moving Pattern forward K[m] characters
            print("Pattern found at index:", i - m + 1, i)
            m = K[m]

    end = time.time()
    print("Time taken to perform Knuth-Morris-Pratt Search:", end - start)
```

Brute-Force Algorithm Implementation

In [80]:
```
def bruteForce(Pattern, Chars):
    start = time.time()
     #get lengths of pattern and chars
    M = len(Pattern)
    N = len(Chars)
    # go through the Pattern[]
    for i in range(N - M + 1):
        j = 0
        # For current index i, check for pattern match
        while(j < M):
            if (Chars[i + j] != Pattern[j]):
                break
            j = j + 1
```

```
        if (j == M):
            print("Pattern found at index: ", i, i + j - 1)

    end = time.time()
    print("Time taken to perform Naive Pattern Search:", end - start)
```

Random 1000 Letter Generation

In [87]:
```
letters = "abc"
Chars = ''.join(random.choice(letters) for i in range(1000))
print(Chars)
Pattern = "abcba"
```

```
acacbbbcccbcacaaabcccacaccaaacaccabaaccaaaabbbbaabbbcabbaaccaacbabcbccccbbcabcacbbbbcbcccbccccabbabbaccacbbbacacbcbcbabcacbbbacbaabbaacccbcbbccccbbcabaccbcbcbccccaabbcbabbcbbaaaccabaabbb
ccaccaabaabababacccccccabaccbccabaccacaacaacaabcbaccacbcacbcbabbaacabcccabccccccbbcccbccaacccbabbcbcbacccbbcbabcaccabbcaccaacabbbccbccabbbcbcabbcbcbaaacbbcabcaabcbbabaacbbbabccaabcaaaccccbac
aaabaabaabaaccccccbabccbcbbbabccccaacbbabacbbaaccabacbbbbabbbaabaaabbaabcccbcaabcccbbaabbcbaabccbaacacbbbbaabaccccccccbcbabcacaabababcbcaacbbbabacccacbcbbcabcbccbabacabcaabababbababacaaaccba
aaaaabacaaaaabcabacbbbcbbbaccbcaccbcabbbbccbcabbcacabcaaccbbabababbcbccaacbbbbbbaabcaaacaaacbbaaaccbcaaabaaaaccbbabccbcabbccabaaccabcacabaaaccaccabccacacbbbccabbbabbaacbacabccacaabbbbbabb
cabbaccbbaabacabcbbaacaacaaabcbbbabaccacbbabbbaccaacaacabaaaccbbcbccbccabcbacbaaccaababcbabaccaabacbaaabcbaacbbcaabbaacbabaaccaaacbabaabbccbabacbccbccbacacbabcaaabbabababbbcacaccbabbbbc
caccacccabbbbabcbabaacabbabaccacaabbcabbbbabbabbcbaabbcccbacabbccbbaba
```

Test

In [88]:
```
def main():
        print("KMP Algorithm Results")
        KMP(Pattern, Chars)
        print("\n")
        print("Brute-Force Algorithm Results")
        bruteForce(Pattern, Chars)
```

In [89]:
```
if __name__ == '__main__':
    main()
```

```
KMP Algorithm Results
Pattern found at index: 228 232
Pattern found at index: 817 821
Pattern found at index: 831 835
Pattern found at index: 848 852
Pattern found at index: 943 947
Time taken to perform Knuth-Morris-Pratt Search: 0.0010294914245605469


Brute-Force Algorithm Results
Pattern found at index:  228 232
Pattern found at index:  817 821
Pattern found at index:  831 835
Pattern found at index:  848 852
Pattern found at index:  943 947
Time taken to perform Naive Pattern Search: 0.0019614696502685547
```