Task 1:
Implementing the KMP Algorithm
Code the Knuth-Morris-Pratt (KMP) algorithm in Java for pattern searching which pre-processes the pattern to reduce the number of comparisons. Explain how this pre-processing improves the search time compared to the naive approach.

ANS:

```java
package Day17;

public class KMPAlgorithm {

    // Method to create the LPS array
    private static int[] computeLPSArray(String pattern) {
        int length = pattern.length();
        int[] lps = new int[length];
        int j = 0;  // Length of the previous longest prefix suffix
        int i = 1;

        // The LPS array is initialized with 0 at the beginning
        lps[0] = 0;

        // Loop to fill the LPS array
        while (i < length) {
            if (pattern.charAt(i) == pattern.charAt(j)) {
                j++;
                lps[i] = j;
                i++;
            } else {
                if (j != 0) {
                    j = lps[j - 1];
                } else {
                    lps[i] = 0;
                    i++;
                }
            }
        }
        return lps;
    }

    // KMP search algorithm
    public static void KMPSearch(String text, String pattern) {
        int textLength = text.length();
        int patternLength = pattern.length();

        // Create the LPS array
        int[] lps = computeLPSArray(pattern);

        int i = 0;  // Index for text
        int j = 0;  // Index for pattern

        while (i < textLength) {
            if (pattern.charAt(j) == text.charAt(i)) {
                i++;
                j++;
            }

            if (j == patternLength) {
                System.out.println("Pattern found at index: " + (i - j));
                j = lps[j - 1];
            } else if (i < textLength && pattern.charAt(j) != text.charAt(i)) {
                if (j != 0) {
                    j = lps[j - 1];
                } else {
                    i++;
                }
            }
        }
    }

    public static void main(String[] args) {
        String text = "ABABDABACDABABCABAB";
        String pattern = "ABABCABAB";
        KMPSearch(text, pattern);
    }
}
```

OUTPUT:

```
Pattern found at index: 10
```