Solution 1

```swift
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

class Program {
    // O(n ^ 2 + m) time | O(n + m) space
    func patternMatcher(_ pattern: String, _ string: String) -> [String] {
        if pattern.count > string.count {
            return []
        }

        let oldPattern = pattern.map { String($0) }
        let newPattern = generateNewPattern(pattern)
        let didSwitch = oldPattern[0] != newPattern[0]

        var counts = ["x": 0, "y": 0]
        let firstYPosition = populateCountsAndGetFirstYPosition(&counts, newPattern)

        if counts["y"] != 0 {
            for lengthOfX in 1 ..< string.count {
                if let countsOfX = counts["x"], let countsOfY = counts["y"] {
                    let lengthOfY: Double = Double(string.count - (lengthOfX * countsOfX)) / Double(countsOfY)

                    if lengthOfY <= 0 || lengthOfY.truncatingRemainder(dividingBy: 1) != 0 {
                        continue
                    }

                    let indexOfY = lengthOfX * firstYPosition

                    let startX = string.index(string.startIndex, offsetBy: 0)
                    let endX = string.index(string.startIndex, offsetBy: lengthOfX)
                    let x = String(string[startX ..< endX])

                    let startY = string.index(string.startIndex, offsetBy: indexOfY)
                    let endY = string.index(string.startIndex, offsetBy: indexOfY + Int(lengthOfY))
                    let y = String(string[startY ..< endY])

                    let potentialMatch = newPattern.map { $0 == "x" ? x : y }.joined(separator: "")

                    if string == potentialMatch {
                        if didSwitch {
                            return [y, x]
                        } else {
                            return [x, y]
                        }
                    }
                }
            }
        } else {
            if let countsOfX = counts["x"] {
                let lengthOfX = string.count / countsOfX
                let startX = string.index(string.startIndex, offsetBy: 0)
                let endX = string.index(string.startIndex, offsetBy: lengthOfX)
                let x = String(string[startX ..< endX])

                let potentialMatch = newPattern.map { $0 == "x" ? x : "" }.joined(separator: "")

                if string == potentialMatch {
                    if didSwitch {
                        return ["", x]
                    } else {
                        return [x, ""]
                    }
                }
            }
        }

        return []
    }

    func generateNewPattern(_ pattern: String) -> [String] {
        let patternCharacters = Array(pattern)

        if patternCharacters[0] == "x" {
            return patternCharacters.map { String($0) }
        } else {
            return patternCharacters.map { $0 == "x" ? "y" : "x" }
        }
    }

    func populateCountsAndGetFirstYPosition(_ counts: inout [String: Int], _ newPattern: [String]) -> Int {
        var firstYPosition = -1

        for (index, currentPatternCharacter) in newPattern.enumerated() {
            if var countPerCharacter = counts[currentPatternCharacter] {
                countPerCharacter += 1
                counts[currentPatternCharacter] = countPerCharacter
            }

            if currentPatternCharacter == "y", firstYPosition == -1 {
                firstYPosition = index
            }
        }

        return firstYPosition
    }
}
```