Solution 1

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

// O(n^2 + m) time | O(n + m) space
function patternMatcher(pattern, string) {
  if (pattern.length > string.length) return [];
  const newPattern = getNewPattern(pattern);
  const didSwitch = newPattern[0] !== pattern[0];
  const counts = {x: 0, y: 0};
  const firstYPos = getCountsAndFirstYPos(newPattern, counts);
  if (counts['y'] !== 0) {
    for (let lenOfX = 1; lenOfX < string.length; lenOfX++) {
      const lenOfY = (string.length - lenOfX * counts['x']) / counts['y'];
      if (lenOfY <= 0 || lenOfY % 1 !== 0) continue;
      const yIdx = firstYPos * lenOfX;
      const x = string.slice(0, lenOfX);
      const y = string.slice(yIdx, yIdx + lenOfY);
      const potentialMatch = newPattern.map(char => (char === 'x' ? x : y));
      if (string === potentialMatch.join('')) {
        return !didSwitch ? [x, y] : [y, x];
      }
    }
  } else {
    let lenOfX = string.length / counts['x'];
    if (lenOfX % 1 === 0) {
      const x = string.slice(0, lenOfX);
      const potentialMatch = newPattern.map(char => (char === 'x' ? x : y));
      if (string === potentialMatch.join('')) {
        return !didSwitch ? [x, ''] : ['', x];
      }
    }
  }
  return [];
}

function getNewPattern(pattern) {
  const patternLetters = pattern.split('');
  if (pattern[0] === 'x') {
    return patternLetters;
  } else {
    return patternLetters.map(char => (char === 'y' ? 'x' : 'y'));
  }
}

function getCountsAndFirstYPos(pattern, counts) {
  let firstYPos = null;
  for (let i = 0; i < pattern.length; i++) {
    const char = pattern[i];
    counts[char]++;
    if (char === 'y' && firstYPos === null) firstYPos = i;
  }
  return firstYPos;
}

exports.patternMatcher = patternMatcher;

```