Solution 1

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

#include <vector>
#include <numeric>
#include <algorithm>
#include <unordered_map>
#include <math.h>
using namespace std;

vector<char> getNewPattern(string pattern);
int getCountsAndFirstYPos(vector<char> pattern,
                          unordered_map<char, int> *counts);

// O(n^2 + m) time | O(n + m) space
vector<string> patternMatcher(string pattern, string str) {
  if (pattern.length() > str.length()) {
    return vector<string>{};
  }
  vector<char> newPattern = getNewPattern(pattern);
  bool didSwitch = newPattern[0] != pattern[0];
  unordered_map<char, int> counts({{'x', 0}, {'y', 0}});
  int firstYPos = getCountsAndFirstYPos(newPattern, &counts);
  if (counts['y'] != 0) {
    for (int lenOfX = 1; lenOfX < str.length(); lenOfX++) {
      double lenOfY =
          ((double)str.length() - (double)lenOfX * (double)counts['x']) /
          (double)counts['y'];
      if (lenOfY <= 0 || fmod(lenOfY, 1) != 0) {
        continue;
      }
      int yIdx = firstYPos * lenOfX;
      string x = str.substr(0, lenOfX);
      string y = str.substr(yIdx, lenOfY);
      vector<string> potentialMatch(newPattern.size(), "");
      transform(newPattern.begin(), newPattern.end(), potentialMatch.begin(),
                [x, y](char c) -> string { return c == 'x' ? x : y; });
      if (str == accumulate(potentialMatch.begin(), potentialMatch.end(),
                            string(""))) {
        return !didSwitch ? vector<string>{x, y} : vector<string>{y, x};
      }
    }
  } else {
    double lenOfX = str.length() / counts['x'];
    if (fmod(lenOfX, 1) == 0) {
      string x = str.substr(0, lenOfX);
      vector<string> potentialMatch(newPattern.size(), "");
      transform(newPattern.begin(), newPattern.end(), potentialMatch.begin(),
                [x](char c) -> string { return x; });
      if (str == accumulate(potentialMatch.begin(), potentialMatch.end(),
                            string(""))) {
        return !didSwitch ? vector<string>{x, ""} : vector<string>{"", x};
      }
    }
  }
  return vector<string>{};
}

vector<char> getNewPattern(string pattern) {
  vector<char> patternLetters(pattern.begin(), pattern.end());
  if (pattern[0] == 'x') {
    return patternLetters;
  } else {
    transform(patternLetters.begin(), patternLetters.end(),
              patternLetters.begin(),
              [](char c) -> char { return c == 'y' ? 'x' : 'y'; });
    return patternLetters;
  }
}

int getCountsAndFirstYPos(vector<char> pattern,
                          unordered_map<char, int> *counts) {
  int firstYPos = -1;
  for (int i = 0; i < pattern.size(); i++) {
    char c = pattern[i];
    counts->at(c)++;
    if (c == 'y' && firstYPos == -1) {
      firstYPos = i;
    }
  }
  return firstYPos;
}

```