Solution 1

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

#include <vector>
#include <numeric>
using namespace std;

vector<vector<int>> getLocations(string str, string subStr);
vector<vector<int>> collapse(vector<vector<int>> locations);
string underscorify(string str, vector<vector<int>> locations);

// O(n*m) | O(n) space
string underscorifySubstring(string str, string subStr) {
  vector<vector<int>> locations = collapse(getLocations(str, subStr));
  return underscorify(str, locations);
}

vector<vector<int>> getLocations(string str, string subStr) {
  vector<vector<int>> locations{};
  int startIdx = 0;
  while (startIdx < str.length()) {
    int nextIdx = str.find(subStr, startIdx);
    if (nextIdx != string::npos) {
      locations.push_back(vector<int>{nextIdx, int(nextIdx + subStr.length())});
      startIdx = nextIdx + 1;
    } else {
      break;
    }
  }
  return locations;
}

vector<vector<int>> collapse(vector<vector<int>> locations) {
  if (locations.empty()) {
    return locations;
  }
  vector<vector<int>> newLocations{locations[0]};
  vector<int> *previous = &newLocations[0];
  for (int i = 1; i < locations.size(); i++) {
    vector<int> *current = &locations[i];
    if (current->at(0) <= previous->at(1)) {
      previous->at(1) = current->at(1);
    } else {
      newLocations.push_back(*current);
      previous = &newLocations[newLocations.size() - 1];
    }
  }
  return newLocations;
}

string underscorify(string str, vector<vector<int>> locations) {
  int locationsIdx = 0;
  int stringIdx = 0;
  bool inBetweenUnderscores = false;
  vector<string> finalChars{};
  int i = 0;
  while (stringIdx < str.length() && locationsIdx < locations.size()) {
    if (stringIdx == locations[locationsIdx][i]) {
      finalChars.push_back("_");
      inBetweenUnderscores = !inBetweenUnderscores;
      if (!inBetweenUnderscores) {
        locationsIdx++;
      }
      i = i == 1 ? 0 : 1;
    }
    string s(1, str[stringIdx]);
    finalChars.push_back(s);
    stringIdx++;
  }
  if (locationsIdx < locations.size()) {
    finalChars.push_back("_");
  } else if (stringIdx < str.length()) {
    finalChars.push_back(str.substr(stringIdx));
  }
  return accumulate(finalChars.begin(), finalChars.end(), string());
}

```