CS106X

Autumn 2015 Cynthia Lee

Section 3 (Week 4) Handout

Problem and solution authors include Marty Stepp, Jerry Cain, Eric Roberts, Ilan Goodman, and Cynthia Lee.

Problem 1: isSubsequence [courtesy of Marty Stepp]

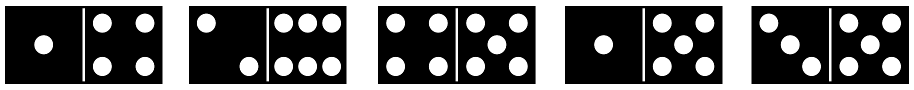
Write a function named `isSubsequence` that takes two strings and returns if the second string is a subsequence of the first string. A string is a subsequence of another if it contains the same letters in the same order, but not necessary consecutively. You can assume both strings are already lowercased.

- `isSubsequence("computer", "core")` returns `false`
- `isSubsequence("computer", "cope")` returns `true`
- `isSubsequence("computer", "computer")` returns `true`

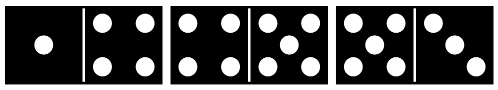
Use this function signature: `bool isSubsequence(string big, string small)`

Problem 2: Domino Chaining [courtesy of Eric Roberts and Jerry Cain]

The game of dominoes is played with rectangular pieces composed of two connected squares, each of which is marked with a certain number of dots. For example, each of the following five rectangles represents a domino:

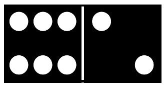


Dominoes are connected end-to-end to form chains, subject to the condition that two dominoes can be linked together only if the numbers match, although it is legal to rotate dominoes 180° so that the numbers are reversed. For example, you could connect the first, third, and fifth dominoes in the above collection to form the following chain:



Note that the 3-5 domino had to be rotated so that it matched up correctly with the 4-5.

Given a set of dominoes, an interesting question to ask is whether it is possible to form a chain starting at one number and ending with another. For example, the example chain shown earlier makes it clear that you can use the original set of five dominoes to build a chain starting with a 1 and ending with a 3. Similarly, if you wanted to build a chain starting with a 6 and ending with a 2, you could do so using only one domino:



On the other hand, there is no way—using just these five dominoes—to build a chain starting with a 1 and ending with a 6.

Dominoes can, of course, be represented in C++ very easily as a pair of integers. Assuming the type **domino** is defined as

```
struct domino {
   int first;
   int second;
};
```

write a predicate function:

```
static bool chainExists(const Vector<domino>& dominoes, int start, int
end);
```

that returns **true** if it is possible to build a chain from **start** to **finish** using any subset of the dominoes in the **dominoes** vector. To simplify the problem, assume that **chainExists** always returns **true** if **start** is equal to **finish**, because you can trivially connect any number to itself with a chain of zero dominoes. (Don't worry about what the chain is—worry only about the yes or not that comes back in the form of a **bool**.) For example, if **dominoes** is the domino set illustrated above, **chainExists** should produce the following results:

```
chainExists(dominoes, 1, 3) → true
chainExists(dominoes, 5, 5) → true
chainExists(dominoes, 1, 6) → false
```

Problem 3: Big O [Cynthia Lee and Ilan Goodman]

These problems are here to start a conversation with your SL, who can explain the answer to each. You are *not* expected to be able to solve all of these right now, and memorizing the details of these is not what this class is about. However, my experience has been that many students in 106X are simply curious about some of these comparisons, so I thought it would be fun (?) to explore some of them in section. TL;DR: in X (as opposed to 106B), we sometimes do stuff just for fun and this is one of those times.

For each pair, say whether $f(n)$ is $O(g(n))$, or $g(n)$ is $O(f(n))$, or both.

a) $f(n) = n \log n + 5n$ $\qquad g(n) = 3n \log n$
b) $f(n) = 10 \log_{10} n$ $\qquad g(n) = 2 \log_2 n$
c) $f(n) = 1.01^n$ $\qquad g(n) = 1000n^3$
d) $f(n) = 1.6^n$ $\qquad g(n) = 2^n$
e) $f(n) = (\log n)^3$ $\qquad g(n) = n^{1/5}$
f) $f(n) = 7n \log(n^2)$ $\qquad g(n) = \begin{cases} 10n^2, & n < 100 \\ 100n, & n \geq 100 \end{cases}$