Computer Science I Program #0: Found in the Shuffle (Strings)

Please check Webcourses for the Due Date

Read all the pages before starting to write your code

An ordinary deck of playing cards contains 52 cards, each of which has a suit and a value. Most card games require that a deck be shuffled at the beginning of a play. However, sometimes the deck isn't shuffled well enough, and there are arrangements of cards in the deck which may cause a bias in the play. Two such arrangements are a ***same-suit sequence*** and an ***ascending sequence.*** A same-suit sequence is simply a sequence of consecutive cards in the deck with the same suit. An ascending sequence is a sequence of consecutive cards in the deck that follow one another in increasing value, with Ace following King and preceding two. Thus, 2S, 5S, KS, 3S AS is a same-suit sequence of length five, and 9C, 10D, JC, QS, KH, AC, 2D is an ascending sequence of length seven, and 2H 3H 4H 5H 6H is both a same-suit and ascending sequence of length five.

The Problem

Given a deck of cards, determine the length of the longest same-suit sequence and the length of the longest ascending sequence.

The Input (to be read from standard input)

The first line will contain a single positive integer, c ($c \leq 25$), representing the number of test cases to process. The test cases follow.

Each test case will be two lines long, each line will contain a string of 52 characters. The first line represents the first 26 cards in the deck and the second line represents the next 26 cards, in order in the deck.

Each card will be represented with two characters, one for its kind and one for its suit, in that order. The 13 characters representing kinds are '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A', in ascending order. The 4 characters representing suits are 'C', 'D', 'H' and 'S'.

The Output (to be printed to standard out)

For each input case, output two space separated integers on a line by themselves: the length of the longest same-suit sequence and the length of the longest ascending sequence, respectively.

Sample Input

```
2
2S5SKS3SAS9CTDJCQSKHAC2D2H3H4H5H6HQD9SJD8HAH4D7CJS8C
KD5C2CQHTS9H5DJHQC4C8D7STHAD7H6D6C6S9D4S7DKC3D8S3CTC
QCTD4C8D7STHAD7H2D3S6D6C6S9D4SAS7D2HKC5H3DTC8S9C3H3C
QD9SQSJD8HAH2SKS4D4H5S7CJS8CKD5C2CACQHJCTS6HKH9H5DJH
```

Sample Output

```
5 7
3 2
```

Implementation Restrictions/ Run-Time/Memory Restrictions

1. You must read the deck for each case into statically allocated string(s).

2. For full credit, your algorithm must run in O(n) time. This means that you can only do a few separate single loops through a deck of cards for full credit. **In particular a correct solution that has a pair of nested loops will NOT receive full credit.**

3. For full credit, you must have appropriately designed functions. **In particular, any correct solution which is fully coded in main will NOT receive full credit.**

3. You must only declare your string variables **INSIDE** your case loop.

Deliverables

You must submit one file over WebCourses:

1) A source file, *deck.c*. Please use stdin, stdout. There will be an automatic 10% deduction if you read input from a file or write output to a file.

2) A file describing your testing strategy, *lastname_Testing.doc(x) or lastname_Testing.pdf*. This document discusses your strategy to create test cases to ensure that your program is working correctly. If you used code to create your test cases, just describe at a high level, what your code does, no need to include it.

3) Files *deck.in* and *deck.out*, storing both the test cases you created AND the corresponding answers, respectively.