ECOO '12 R1 P2 - Decoding DNA

Time Limit: 2.0s   **Memory Limit:** 64M

DNA is made up of two twisted strands that encode genes using long combinations of four bases: Adenine, Cytosine, Guanine and Thymine. The strands are complementary to one another, meaning that Adenine and Thymine are always opposite each other, and Cytosine and Guanine are always opposite each other, like this.

A double strand of DNA:

> ATCAAGGCCTATTCCGGGAAAGG
> TAGTTCCGGATAAGGCCCTTTCC

In order for the information in a gene to be used, it has to be transcribed into a strand of RNA. During this process, a portion of one strand of DNA is transcribed – this portion is known as the transcription unit. The start of the sequence to be transcribed is signaled by a sequence of bases known as a promoter, and the end is signaled by a sequence known as the terminator. For our purposes, the promoter is the sequence `TATAAT`, which begins 10 bases before the start of the transcription unit, and the terminator consists of two distinct, complementary, reversed sequences of at least length 6 that cause the RNA molecule to coil back on itself and pinch off the transcribed strand. If `TATAAT` appears twice on a strand, only the first occurrence counts as the promoter. An example is shown below.

> AGATTA**TATAAT**GATA<u>GGATTTAGATTGACCC</u>**GTCATGCA**AGTCCA**TGCATGAC**AGC

In this example the promoter and terminator sequences are boldfaced, and the transcription unit is underlined. The resulting RNA will be complementary to the transcription unit, except that in RNA Uracil takes the place of Thymine. For this example, the result looks like this:

> CCUAAAUCUAACUGGG

The input will contain five single strands of DNA, one on each line. Write a program to output the RNA sequence that results from the transcription process. The sequences should be numbered starting at 1, with a colon and a single space character following the number, as shown below.

Sample Input

```
AGATTATATAATGATAGGATTTAGATTGACCCGTCATGCAAGTCCATGCATGACAGC
AGTCTTCAAGGGGATTCCCAGGTATATAATGCAGATCGCGACGAAATATCGGGCGGGATCCATACCGACCCAGCCGCCCGA
TATAATGGGGGAGAGACCGAGTGTTTAAGTCCCGAGGGATCGGGAGTGAGATTGAGGGAATTCCGGGAATCTCACT
```

Sample Output

```
1: CCUAAAUCUAACUGGG
2: UAGCGCUGCUUUAU
3: CUCUCUGGCUCACAAAUUC
```