**Multiple Alignment Problem**

*Find an alignment of three strings.*

**Input:** Strings *r*, *s*, and *t*.

**Output:** The maximum score of a multiple alignment of these three strings, followed by a multiple alignment of the three strings achieving this maximum using a scoring function in which the score of an alignment column is 1 if all three symbols are identical and 0 otherwise.

You are a Bioinformatician and you have obtained three DNA sequences. You want to line up the amino acids of the three sequences to find positions at which they are similar.  The score of an alignment is defined as the sum of the scores of each position of the alignment, where the score of a position *i* is 1 if all 3 characters match, otherwise 0.

**Input Format.** The first line of the input contains a string *r*, the second line of the input contains a string *s*, and the third line of the input contains a string *t*.

**Output Format.** The first line of the output should contain the maximum score of an alignment between the three input strings, and the next three lines should contain an alignment achieving this maximum score. Specifically, the second line should contain *r* with gaps placed appropriately, the third line should contain *s* with gaps placed appropriately, and the fourth line should contain *t* with gaps placed appropriately.

**Constraints.** |*r*| ≤ 10; |*s*| ≤ 10; |*t*| ≤ 10

**SAMPLE DATASET:**

Input:

ATATCGG

TCCGA

ATGTACTG

Output:

3

ATATCC-G-

---TCC-GA

ATGTACTG-

The highest score of an alignment between ATATCCG, TCCGA, and ATGTACTG is 3, and the above alignment achieves this maximum score.

**TEST DATASET 1:**

Input:

A

AT

A

Output:

1

A-

AT

A-

This test makes sure that your code follows the scoring scheme of the problem. The score will only increase if there is a match between all three strings. Otherwise the score remains unchanged. In this dataset your code should not penalize the indels in reconstructed alignment for strings *r* and *t*.

**TEST DATASET 2:**

Input:

AAAAT

CCCCT

T

Output:

1

AAAAT

CCCCT

----T

This test makes sure that your code can accurately reconstruct the alignment when one of the aligned strings is primarily indels. When backtracking to reconstruct the alignment in this problem be sure to handle all the base cases. In regular two-string alignment you only need to worry about backtracking base cases for the very first row and the very first column of the dynamic programming matrix. In three-string alignment you also need to consider the two dimensional matrices formed by every possible pair of strings. These two dimensional matrices also need to be considered when backtracking.

**TEST DATASET 3:**

Input:

AT

ACCT

AGGGGT

Output:

2

A------T

A----CCT

AGGGG--T

This test makes sure that your code correctly forces a three character match whenever possible. The reconstructed alignment is not set in stone, just make sure that the first character and last character of the three strings align with each other. Note that in the example output extra indels are introduced when not necessary (mismatching CC and GG doesn’t have a penalty); the scoring scheme in this problem allows such an alignment. Make sure that your code does not unnecessarily punish extra indels.

**TEST DATASET 4:**

Input:

GGAG

TT

CCCC

Output:

0

----GGAG

--TT----

CCCC----

This test makes sure that your code is able to output a score of zero if there are no matching characters between the three strings. For this dataset any alignment reconstruction that includes all characters from each string is acceptable. No matter how the three strings in this dataset are aligned there is no way to obtain a non-zero score. If your output score doesn’t match the correct output score make sure that your implementation doesn’t disallow a score of zero when necessary.

**TEST DATASET 5:**

Input:

T

T

T

Output:

1

T

T

T

This test makes sure that your code is able to correctly handle inputs in which all three strings are one character long. Since all the strings are the same in this dataset the output will have a score of one and the reconstructed alignment will be the same as the input. If your output doesn’t match the correct output it’s likely that your implementation has an error in reconstruction. If your code outputs extraneous gap characters in the reconstructed alignment there is likely an error in the termination of your reconstruction. Double check your base cases.