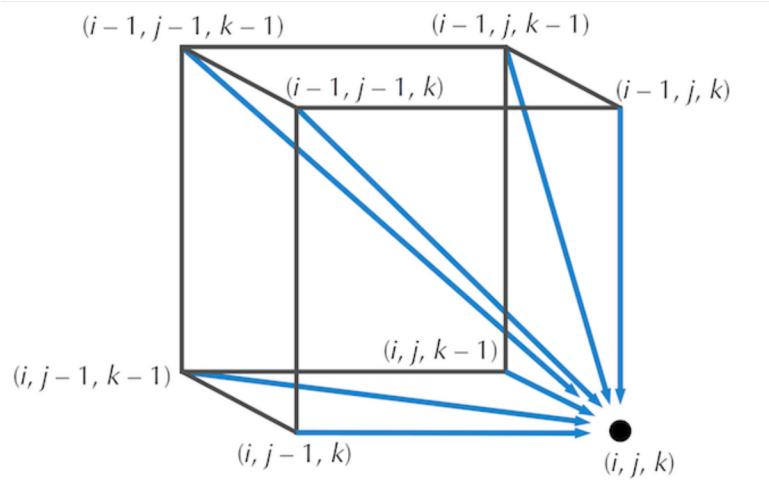
5M Find a Highest-Scoring Multiple Sequence Alignment

Multiple Sequence Alignment Problem
Find a highest-scoring alignment of multiple sequences.

Input: Three DNA strings and a scoring parameters.
Output: A highest-scoring multiple alignment of these strings (with respect to the scoring parameters) and its score.




```
ATATCC-G-
---TTC-GA
ATGTACTG-
```

Formatting

Input: Three space-separated DNA strings x, y, and z.
Output: The maximum multiple sequence alignment score of x, y, and z as an integer followed by a newline-separated multiple sequence alignment of x, y, and z achieving this maximum score (if more than one multiple sequence alignments achieving the maximum score exist, you may return any one). Use a simple scoring function in which the score of an alignment column is equal to 1 if all of the column's symbols are identical, and 0 if even one symbol disagrees.

Constraints

- The lengths of x, y, and z will be between 1 and 10^3.
- Strings x, y, and z will be DNA strings.

Test Cases

Case 1

Description: The sample dataset is not actually run on your code.

Input:

```
ATATCGG
TCCGA
ATGTACTG
```

Output:

```
3
ATATCC-G-
---TCC-GA
ATGTACTG-
```

Case 2

Description: 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 x and z.

Input:

```
A
AT
A
```

Output:

```
1
A-
AT
A-
```

Case 3

Description: 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.

Input:

```
AAAAT
CCCCT
T
```

Output:

```
1
AAAAT
CCCCT
----T
```

Case 4

Description: 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.

Input:

```
AT
ACCT
AGGGGT
```

Output:

```
2
A------T
A----CCT
AGGGG--T
```

Case 5

Description: 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.

Input:

```
GGAG
TT
CCCC
```

Output:

```
0
----GGAG
--TT----
CCCC----
```

Case 6

Description: 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.

Input:

```
T
T
T
```

Output:

```
1
T
T
T
```

Case 7

Description: A larger dataset of the same size as that provided by the randomized autograder. Check input/output folders for this dataset.