rotating

Swanky Shen has recently learnt a very powerful string operation: rotation! What it does is: Given a string S, we can rotate it by shifting some (possibly empty) prefix of S to the back of S. For instance, `abca` has rotations `abca`, `bcaa`, `caab` and `aabc`.

With this new knowledge, he is interested to solve the well-known Longest Common Subsequence (LCS) problem again! That is, given two strings S and T, what is the length of the longest common subsequence (not necessarily contiguous) between some rotation of S and some rotation of T?

However, he is stuck. Can you help him?

Implementation

You should submit a file that implements the following procedure:

```
int rotating(int N, int M, char S[], char T[])
```

It should return the length of the longest common subsequence of the two given strings S and T after some rotation of both of them.

Your procedure: rotating

```
int rotating(int N, int M, char S[], char T[])
```

Description

You will need to implement this function.

It should return the length of the longest common subsequence of the two given strings S and T after some rotation of both of them.

Parameters

- `int N` - The length of string S
- `int M` - The length of string T
- `char S[]` - The string S of length N
- `char T[]` - The string T of length M

Return value

Return the length of the longest common subsequence of the two given strings S and T after some rotation of both of them.

Subtasks

S and T will only contain the characters a to z.

Subtask 1 (11 points)

$1 \leq N, M \leq 100$.

Subtask 2 (24 points)

$1 \leq N, M \leq 500$.

Subtask 3 (65 points)

$1 \leq N, M \leq 2000$.

Sample Input and Output

Input	Output
4 4 cbca bacd	3

One possible to way to obtain a length of 3 is to rotate S to `cacb` and T to `acdb`. The longest common subsequence will then be `acb`.