String Test

Time limit: 1.0s **Memory limit:** 256M

Given two strings a and b, support the following four operations:

1. Set the yth character of a to z and then output $F(a, b)$.
2. Consider the 1-indexed substring of a starting at index y ending at index z, call it s. Print $F(s, b)$.
3. Consider the suffixes of b starting with the yth and zth indexed characters, call them t and u. Print $f(t, u)$.
4. Consider the 1-indexed substrings of b starting and ending at l_1 and r_1, and l_2 and r_2. Determine if their concatenation is a substring of b. Print `yes` if so, print `no` otherwise.

$f(x, y)$ is the length of the longest common prefix of x and y.

$F(x, y)$ is a pair (p, q) where p is the maximum value of $f(z, y)$ where z varies over all suffixes of x, and q is the number of suffixes that yield that maximum value.

Constraints

$N, M, Q \leq 10^5$

Each character in the string will be a positive integer no larger than 10^5.

Input

The first line contains a single integer, the test case number. You may ignore this line.

The next line contains a single positive integer, N, representing the length of a.

The next line contains N space-separated integers, the string a.

The next line contains a single positive integer, M, representing the length of b.

The next line contains M space-separated integers, the string b.

The next line contains a single positive integer, Q, representing the number of operations.

The next Q lines indicate the operations to be performed in order.

If the operation is the first one, there will be three space-separated integers, 1, y, and z.

If the operation is the second one, there will be three space-separated integers, 2, y, and z.

If the operation is the third one, there will be three space-separated integers, 3, y, and z.

If the operation is the fourth one, there will be five space-separated integers, 4, l_1, r_1, l_2, and r_2.

Output

Print an answer for each operation. All answers go on their own lines.

Sample Input

```
0
10
1 2 3 3 3 1 2 3 2 1
3
1 3 1
10
3 1 3
4 3 3 2 2
2 2 10
1 3 2
2 7 9
2 7 10
2 3 9
2 2 8
1 7 1
1 4 2
```

Sample Output

```
1
yes
1 2
1 3
0 3
1 1
1 1
1 1
2 1
2 1
```