Problem B. The Matching System

As the leader of the safety department, you are asked to check an ancient matching system in your company.

The system is fed with two strings, a to-be-matched string and a pattern string, and will determine whether the former can match the latter. The former string is a strict binary string(i.e., contains only **0** and **1**), and the latter string consists of four types of characters **0**, **1**, *, ^, where * means match zero or more arbitrary binary characters, and ^ means match exactly one binary character.

The system has two matching methods: maximum matching and minimum matching.

Consider the starting positions of the two strings. The maximum matching method will make different decisions based on the current character of the pattern string:

- *: The system will enumerate i from L to 0, where L is the remaining length of the to-be-matched string. Before each enumeration starts, the system consumes 1 unit of energy. Then it temporarily assumes that the current * in the pattern string matches the consecutive i characters in the to-be-matched string, and tries to match the remaining positions of two strings recursively. As long as one attempt is successful, the system will give up the remaining enumeration and stop the whole system. Otherwise, it will try the next enumeration until all attempts are tried and finally return to the previous * enumeration.
- **0,1**: The system will stop and return to the previous * enumeration if the to-be-matched string has been exhausted. Otherwise, it consumes 1 unit of energy to compare the current characters between the pattern string and the to-be-matched string. It will continue analyzing the remaining positions of these two strings if the result is the same, otherwise, return back to the previous * enumeration.
- ^: The system will stop and return to the previous * enumeration if the to-be-matched string has been exhausted. Otherwise, it consumes 1 unit of energy and moves on of two strings.

When the pattern string is exhausted, the system will check the to-be-matched string at the same time. It will return"Yes" and stop the whole process if the to-be-matched string is also exhausted, otherwise, it will return to the previous * enumeration. After all attempts are tried and no matching method is found, the system will eventually return "No".

Minimum matching does a similar thing except for the enumeration order of * (i.e., enumerate i from 0 to L).

These two matching methods seem not very effective, so you want to hack them. Please construct both a pattern string and a to-be-matched string of length n for each matching method, so that the system answers "Yes" and the energy consumption is as large as possible.

Input

There is only one test case in each test file.

The first and only line contains an integer n ($1 \leq n \leq 10^3$) indicating the length of strings need to be constructed.

Output

Please output the pattern string, the to-be-matched string, and the energy cost for the maximum matching method in the first 3 lines. Then output the pattern string, the to-be-matched string, and the energy cost for the minimum matching method in the next 3 lines.

If there are multiple constructing ways, you can output any of them.

The energy cost may be very large, so you need to output the value modulo $(10^9 + 7)$. Note that this is only for your convenience and you need to maximize the energy cost before the modulus.

Example

standard input	standard output
3	*0* 011 8 **1 101 7