Combo

You are playing an action video game. The game controller has 4 buttons, `A`, `B`, `X`, and `Y`. In this game, you can get coins with combo moves. You can make a combo move by pressing buttons in sequence.

This game has a secret sequence of buttons, which can be represented as a string S of those 4 characters. You don't know the string S, but you know its length N.

You also know that the first character of S never reappears in it. For example, S can be `"ABXYY"` or `"XYYAA"`, but cannot be `"AAAAA"` or `"BXYBX"`.

You can press a sequence of up to $4N$ buttons for a combo move. Let p be the string which represents the sequence of the buttons you pressed. The number of coins you get for this move is calculated as the length of the longest prefix of S which is also a substring of p. A substring of a string t is a contiguous (possibly empty) sequence of characters within t. A prefix of t is a substring of t that is empty or contains the first character of t.

For example, if S is `"ABXYY"` and p is `"XXYYABYABXAY"`, you will get 3 coins because `"ABX"` is the longest prefix of S that is also a substring of p.

Your task is to determine the secret string S using few combo moves.

Implementation details

You should implement the following function:

```
string guess_sequence(int N)
```

- `N`: the length of string S.
- This function is called exactly once for each test case.
- This function should return the string S.

Your program can call the following function:

```
int press(string p)
```

- `p`: a sequence of buttons you press.
- `p` must be a string of length between 0 and $4N$, inclusive. Each character of `p`

must be `A`, `B`, `X`, or `Y`.
- You cannot call this function more than $8\,000$ times for each test case.
- This function returns the number of coins you get when you press the sequence of buttons represented by `p`.

If some of the above conditions are not satisfied, your program is judged as **`Wrong Answer`**. Otherwise, your program is judged as **`Accepted`** and your score is calculated by the number of calls to `press` (see Subtasks).

Example

Let S be `"ABXYY"`. The grader calls `guess_sequence(5)`. An example of communication is shown below.

Call	Return
`press("XXYYABYABXAY")`	3
`press("ABXYY")`	5
`press("ABXYYABXYY")`	5
`press("")`	0
`press("X")`	0
`press("BXYY")`	0
`press("YYXBA")`	1
`press("AY")`	1

For the first call to `press`, `"ABX"` appears in `"XXYYABYABXAY"` as a substring but `"ABXY"` does not, so 3 is returned.

For the third call to `press`, `"ABXYY"` itself appears in `"ABXYYABXYY"` as a substring, so 5 is returned.

For the sixth call to `press`, no prefix of `"ABXYY"` but the empty string appears in `"BXYY"` as a substring, so 0 is returned.

Finally, `guess_sequence(5)` should return `"ABXYY"`.

The file `sample-01-in.txt` in the zipped attachment package corresponds to this example.

Constraints

- $1 \leq N \leq 2\,000$
- Each character of the string S is `A`, `B`, `X`, or `Y`.

- The first character of S never reappears in S.

In this problem, the grader is NOT adaptive. This means that S is fixed at the beginning of the running of the grader and it does not depend on the queries asked by your solution.

Subtasks

1. (5 points) $N = 3$
2. (95 points) No additional constraints. For this subtask, your score for each test case is calculated as follows. Let q be the number of calls to `press`.
   - If $q \leq N + 2$, your score is 95.
   - If $N + 2 < q \leq N + 10$, your score is $95 - 3(q - N - 2)$.
   - If $N + 10 < q \leq 2N + 1$, your score is 25.
   - If $\max\{N + 10, 2N + 1\} < q \leq 4N$, your score is 5.
   - Otherwise, your score is 0.

Note that your score for each subtask is the minimum of the scores for the test cases in the subtask.

Sample grader

The sample grader reads the input in the following format:

- line 1: S

If your program is judged as **`Accepted`**, the sample grader prints `Accepted: q` with `q` being the number of calls to the function `press`.

If your program is judged as **`Wrong Answer`**, it prints `Wrong Answer: MSG`. The meaning of `MSG` is as follows:

- `invalid press`: A value of `p` given to `press` is invalid. Namely, the length of `p` is not between 0 and $4N$, inclusive, or some character of `p` is not `A`, `B`, `X`, or `Y`.
- `too many moves`: The function `press` is called more than $8\,000$ times.
- `wrong guess`: The return value of `guess_sequence` is not S.