Immortal Grasshopper

Mehmet, as someone who loves computer games, has been playing Immortal Grasshopper for the last 10 hours. Immortal Grasshopper is a computer game where you control your character, a karate master grasshopper, with buttons. The opponent you face is a computer-controlled ant.

The grasshopper can perform a total of S ($1 \leq S \leq 4$) different attacks. These different attacks are named with the first S letters of the alphabet. For example:

- $S = 2$: attacks are named A, B.
- $S = 4$: attacks are named A, B, C, D.

Each attack that **hits the opponent** deals 1 damage.

Although you know the rules of the game, since Mehmet fell asleep after playing for 10 hours, there is no one to teach you the controls. You found a controller with N buttons on the table (The buttons are numbered from 0 to $N - 1$). After playing for a while, you realized that each of the buttons can perform one of the S different attacks, but you do not know which button does what. Luckily, thanks to the sticker on the very first button, you see that it performs an A-type attack. Furthermore, you know that **each different type of attack is assigned to at least one button.**

The task given to you is to determine what type of attacks all the buttons on the controller perform by playing this game.

You play the game as described below. To attack the opponent ant, you use consecutive button combinations called combos. A combo works as follows:

- You create a combo by pressing any number of buttons in sequence.
- You can press the buttons in any order within a combo.
- The **same button** can be pressed multiple times in a combo.
- A combo can have a maximum length of 400.

The game takes your moves and ultimately shows you the total damage you've dealt.

The amount of damage a combo deals to the ant is calculated as follows. Let $t(X)$ denote the attack that comes alphabetically right after attack X. For the attack represented by the last character, the $t(X)$ value is defined as the A attack. For example:

- For $S = 4$: $t(A) = B$, $t(B) = C$, $t(C) = D$, $t(D) = A$.
- For $S = 2$: $t(A) = B$, $t(B) = A$.

The opponent ant can counter some of your attacks using sneaky tactics. More specifically, if you used attack X as your last **successful (unblocked)** attack that dealt damage, and you use the $t(X)$ attack as your next attack, the ant will block your $t(X)$ attack and you will deal no damage. If it cannot block, you will deal 1 damage to the ant (The first attack always deals damage).

Your goal is to find out and report which attacks all the buttons perform by using the minimum total number of combos.

Implementation Details

You are expected to implement the following function:

```
std::string olumsuz_cekirge(int N, int S)
```

- The value N represents the number of buttons on the controller.
- The value S represents the number of different attack types in the game.
- This function should return a `string` of length N, representing the attacks assigned to the buttons.

While doing this, you can use the `kombo` function below:

```
int kombo(std::vector<int> buton)
```

- You can call the `kombo` function at most 400 times during each `olumsuz_cekirge` call.
- The `buton` array specifies the indices of the buttons you want to use in the combo.
- The values of the `buton` array must be integers from 0 to $N - 1$.
- The length of the `buton` array can be at most 400.

You are expected to solve T independent scenarios in each test with the same N and S values. For each independent scenario, `olumsuz_cekirge` function will be called once.

Constraints

- $1 \leq T \leq 100$
- $4 \leq N \leq 400$
- $1 \leq S \leq 4$
- **The grader is not adaptive.** More specifically, the input is fixed before the `olumsuz_cekirge` function is called and does not change during operations.

Subtasks & Scoring

The score obtained from the i-th subtask will be 0 if the `kombo` function is called with parameters that do not meet the criteria above in any scenario, or if the `olumsuz_cekirge` function returns an incorrect result. Otherwise, it will be equal to the following formula:

$$P_i \times \min\left(1, \left(\frac{400 - q}{400 - Q_i}\right)^3\right)$$

- The value P_i indicates the total possible points that can be obtained from the i-th subtask.
- The value q indicates the maximum number of times the `kombo` function is called among the scenarios across all tests in the i-th subtask.
- The Q_i values are given in the subtask descriptions.

Subtasks	Points	Additional Constraints	Dependencies
1	1	$S = 1$, $Q_1 = 0$, $P_1 = 1$	—
2	15	$S = 2$, $Q_2 = 50$, $P_2 = 15$	—
3	40	$S = 3$, $Q_3 = 80$, $P_3 = 40$	—
4	44	$S = 4$, $Q_4 = 103$, $P_4 = 44$	—

Examples

```
olumsuz_cekirge(6, 3)
```

In the examples, the attacks corresponding to each button are `ABCBBA` in order, so ithis function should return this `string`.

```
kombo([0, 1, 2])
```

The call above returns 2. The attacks in this combo would be ABC. The first A attack hits. The B attack is blocked. Since the last attack that dealt damage was A, the ant cannot block the C attack and takes damage.

```
kombo([0, 0, 0])
```

The call above returns 3. The attacks in this combo would be AAA. All A attacks deal damage.

```
kombo([5, 1, 3, 2, 2, 0])
```

The call above returns 3. The attacks in this combo would be $ABBCCA$. The first A attack hits. The B attack is blocked. Since the last attack that hit is still A, the second B attack is also blocked. The first and second C attacks deal damage. Since the last attack that hit is C, the opponent ant expects A, therefore the final A attack is blocked.

```
kombo([2, 2, 1])
```

The call above returns 3. The attacks in this combo would be CCB. All attacks deal damage.

Sample Grader

The sample grader reads the inputs in the following format where C_i is a `string` of size N representing the secret attacks of the buttons for the i-th scenario:

```
T N S
C_1
C_2
...
C_T
```

It will print the following for each scenario: If the scenario is solved correctly, the total number of operations used is printed. If there is any error, one of the following error messages is printed:

```
Yanlis cevap (Wrong answer)
Buton asildi (Button limit exceeded)
Yanlis buton (Invalid button)
Kombo asildi (Combo limit exceeded)
```

- `Yanlis cevap (Wrong answer)`: Printed if there is a mistake in the answer of any button.
- `Buton asildi (Button limit exceeded)`: Returned if more than 400 buttons are used in a single `kombo` function call.
- `Yanlis buton (Invalid button)`: Returned if one of the button indices is not a number between 0 and $N-1$ in a `kombo` function call.
- `Kombo asildi (Combo limit exceeded)`: Returned if the `kombo` function is called more than 400 times during a single `olumsuz_cekirge` function call.