Problem

Go, go, Power Arrangers! Everyone loves this team of five superhero high school students who wear the letters A, B C, D, and E. When they stand side by side to confront evil monsters, they arrange their team in one of 120 possible ifferent left-to-right orders, giving them various different tactical superpowers. They are even more popular than the eenage Permutant Ninja Turtles!

Some critics of the show claim that the team only has its arrangment gimmick so that the owners of the show can sel 120 separate sets of 5 action figures, each of which features the team in a different left-to-right order, glued to a bas so that the set cannot be rearranged. As an avid Power Arrangers fan, you have collected 119 of these sets, but you o not remember which set you are missing. Your 119 sets are lined up horizontally along a shelf, such that there are total of $119 \times 5 = 595$ action figures in left-to-right order. You do not remember how the sets are arranged, but you now that the permutation of the sets is selected uniformly at random from all possible permutations, and independent y for each case.

You do not want to waste any time figuring out which set you are missing, so you plan to look at the letters on at mo t F figures on the shelf. For instance, you might choose to look at the letter on the eighth figure from the left, which ould be the third figure from the left in the second set from the left. When looking at a figure, you only get the letter rom that one figure; the letters are hard to see, and the different team members look very similar otherwise!

After checking at most F figures, you must figure out which of the sets is missing, so you can complete your collecti n and be ready to face any possible evil threat!

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems sect on of our FAQ.

Initially, your program should read a single line containing two integers T, the number of test cases, and F, the numb r of figures you are allowed to inspect per test case. Then, you need to process T test cases.

Within each test case, the missing set of figures is chosen uniformly at random from all possible sets, and the order o the remaining sets is chosen uniformly at random from all possible orders as well. Every choice is made independen ly of all other choices and of your inputs.

In each test case, your program will process up to F + 1 exchanges with our judge. You may make up to F exchange of the following form:

Your program outputs one line containing a single integer between 1 and 595, inclusive, indicating which figure (in l ft-to-right order along the shelf) you wish to look at. As a further example, 589 would represent the fourth figure fro the left in the second set from the right.

The judge responds with one line containing a single uppercase letter A, B, C, D, or E, indicating the letter on that fi ure. If you sent invalid data (e.g., a number out of range, or a malformed line), the judge will instead respond with o e line containing the single uppercase letter N.

Then, after you have made as many of the F exchanges above as you want, you must make one more exchange of th following form:

Your program outputs one line containing a single string of five uppercase letters: the permutation corresponding to he missing set (e.g., CADBE).

The judge responds with one line containing a single uppercase letter: Y if your answer was correct, and N if it was ot (or you provided a malformed line). If you receive Y, you should begin the next test case, or stop sending input if here are no more test cases.

After the judge sends N to your input stream (because of either invalid data or an incorrect answer), it will not send a y other output. If your program continues to wait for the judge after receiving N, your program will time out, resulti

g in a Time Limit Exceeded error. Notice that it is your responsibility to have your program exit in time to receive a rong Answer judgment instead of a Time Limit Exceeded error. As usual, if the memory limit is exceeded, or your p ogram gets a runtime error, you will receive the appropriate judgment.

Limits
$1 \leq T \leq 50$.
Time limit: 40 seconds per test set.
Memory limit: 1GB.
The missing set, and the order of the remaining sets, are chosen uniformly and independently at random.

Test set 1 (Visible)
$F = 475$.

Test set 2 (Hidden)
$F = 150$.

Testing Tool
You can use this testing tool to test locally or on our servers. To test locally, you will need to run the tool in parallel ith your code; you can use our interactive runner for that. For more information, read the Interactive Problems sectio of the FAQ.

Local Testing Tool
To better facilitate local testing, we provide you the following script. Instructions are included inside. You are encou aged to add more test cases for better testing. Please be advised that although the testing tool is intended to simulate he judging system, it is NOT the real judging system and might behave differently.

If your code passes the testing tool but fails the real judge, please check the Coding section of our FAQ to make sure that you are using the same compiler as us.

Sample Interaction
This interaction corresponds to Test set 1.

```
t, f = readline_int_list()   // Reads 50 into t and 475 into f
printline 10 to stdout       // Looks at the last figure in the second set
                             // from the left
flush stdout
n = readline_string()        // Reads B into n. Ooh, team member B! They may
                             // not have the leadership ability of A, or the
                             // technical skill of C, but they entertain the
                             // team with clever quips!
printline 11 to stdout       // Looks at the first figure in the third set
                             // from the left
flush stdout
n = readline_string()        // Reads B into n. Notice that B is at the start
                             // of the third set, whereas they were at the
                             // end of the second set.
printline 14 to stdout       // Looks at the fourth figure in the third set
                             // from the left
flush stdout
n = readline_string()        // Reads D into n. Silent and brooding, team
                             // member D nonetheless fights fiercely to
```

```
                        // protect their friends... and the world!
  printline ABCDE to stdout    // We foolishly make a wild guess even though we
                        // could have looked at up to 472 more figures.
  flush stdout
  verdict = readline_string()  // Reads N into verdict (judge has decided our
                        //   solution is incorrect)
  exit                   // Exits to avoid an ambiguous TLE error
```

Solution:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

int main(){
 ios_base::sync_with_stdio(false);
 int num_cases, F;
 cin >> num_cases >> F;
 for(int case_num = 1; case_num <= num_cases; ++case_num){
  const int n = 119;
  vector<int> candidates;
  for(int i = 0; i < n; ++i){ candidates.push_back(i); }
  vector<char> solution;
  for(int offset = 0; offset < 4; ++offset){
   vector<vector<int>> next_candidates(5);
   for(const auto index : candidates){
    cout << (index * 5 + offset + 1) << endl << flush;
    string name;
    cin >> name;
    next_candidates[name[0] - 'A'].push_back(index);
   }
   int min_count = n;
   for(int i = 0; i < 5; ++i){
    if(count(solution.begin(), solution.end(), 'A' + i) > 0){ continue; }
    min_count = min<int>(min_count, next_candidates[i].size());
   }
   for(int i = 0; i < 5; ++i){
    if(count(solution.begin(), solution.end(), 'A' + i) > 0){ continue; }
    if(next_candidates[i].size() == min_count){
     solution.push_back('A' + i);
     candidates = next_candidates[i];
    }
   }
  }
  for(int i = 0; i < 5; ++i){
   if(count(solution.begin(), solution.end(), 'A' + i) > 0){ continue; }
   solution.push_back('A' + i);
  }
```

```
  solution.push_back('\0');
  cout << solution.data() << endl << flush;
  string status;
  cin >> status;
 }
 return 0;
}
```