Your solution to this problem must be implemented using an object-oriented approach with multiple classes. Any other solution will receive zero points, even if the testing system deems it to be correct.

The genre of trick-taking card games encompasses many popular games that revolve around each player having some number of cards (which may be dealt all at once, or as the game progresses), and a number of rounds (or "tricks") where each player plays a card. Based on these cards, and specially on how they compare to each other, players receive a number of points.

In this problem, you will write a program to determine the winner of a trick-taking game. This game involves P players (numbered from 1 to P), and we use a deck with four suits (`A`, `B`, `C`, `D`) and 10 values (1 through 10). In other words, there are a total of 40 cards in the deck:

```
A 1
A 2
A 3
...
D 9
D 10
```

At the start of the game, one of the suits is selected as the *trump suit* (in the sense that it trumps, or beats, all other suits). The trump suit remains the same for the entire game, and the manner in which this suit is selected is not relevant to the problem.

In each round, each player takes a card from the deck and immediately plays that card. In the first round of the game, player 1 draws the first card, followed by player 2, then player 3, etc. until player P draws a card. Later on we will see this order will change in subsequent rounds of the game.

The suit of the first card drawn in a round is designated as the *trick suit* (note that it is possible for the trick suit and the trump suit to be the same in a given round). In each round, there are P cards in play, and the winning card is selected based on the following rules:

- Cards with a suit other than the trump suit or the trick suit always lose.
- The trump suit always beats the trick suit, regardless of value.
- When two cards have the same suit, the highest value card wins.

For example, consider a game with four players where the trump suit is `A`. In the first round, the players draw and play the following cards:

- Player 1 draws `D 4`
- Player 2 draws `C 8`
- Player 3 draws `D 10`
- Player 4 draws `A 3`

The suit of the first card, `D`, is designated as the trick suit. That means that cards in suits other than `D` (the trick suit) or `A` (the trump suit) lose automatically. So, only cards `D 4`, `D 10`, and `A 3` are in play and, since `A 3` has the trump suit, it wins over `D 10` and `D 4`, despite having a lower number value than those cards. This means that player 4 wins this round.

If, on the other hand, the trump suit were `B`, then `D 10` (held by player 3) would win the round, since `C 8` and `A 3` lose automatically, and `D 10` has a higher number value than `D 4`.

A full game involves using a deck of N cards (out of the 40 possible cards), and playing rounds until there are fewer than P cards in the deck. Since the first player to draw a card has an advantage over the other players (because their card will determine the trick suit), the order in which players draw the cards changes in each round. More specifically, if player p drew the first card in a given round, then player $p + 1$ will draw the first card in the next round (followed by the other players in increasing order, wrapping around to 1). For example, in the second round of a four-player game, player 2 would draw the first card, followed by players 3, 4, and 1. Then, in the third round, player 3 would draw the first card, followed by players 4, 1, and 2.

In a given round, the player with the winning card collects all the cards in that round, and earns points based on the number value of the cards (regardless of whether the cards were part of the trick or trump suit in that round). The points are determined based on this table:

Card Number	Points
10	10
9	9
8	4
7	3
6	2
1, 2, 3, 4, 5	0

For example, in the round shown earlier, the total value of the cards in that round was 14:

Card	Points
`D 4`	0
`C 8`	4
`D 10`	10
`A 3`	0
Total	14

The player with the most points wins the game; if two or more players have the

same number of points by the end of the game, then the player with the *lowest* number wins.

For example, consider a game played with eight cards and four players, and where the trump suit is `B`.

- Round 1
  - Player 1 draws `D 4`
  - Player 2 draws `C 8`
  - Player 3 draws `D 10`
  - Player 4 draws `A 3`
- Round 2
  - Player 2 draws `D 7`
  - Player 3 draws `A 4`
  - Player 4 draws `D 6`
  - Player 1 draws `B 5`

Player 3 (holding card `D 10`) wins the first round, earning 14 points, and player 1 (holding card `B 5`) wins the second round, earning 5 points. So, player 3 wins the game with 14 points.

In this problem, you will be given the specification of a game, and must compute who would win the game (and with how many points) with each of the possible suits as the trump suit. In other words, you will play the same game four times, setting a different trump suit each time.

You must solve the problem in **one** of Python 3, Java, or C++. Unlike other problems, you are not provided with any starter code, and must implement your solution from scratch. You will be reading the input from a file, and printing the output to the terminal (i.e., standard output). The exact specification of the input and output can be found further below.

All of your code for this task will go into a single file. A note for Java programmers: you can include multiple classes in file as long as all but one of the classes are private.

You **must** use one of the following file names for the file:

- **Python 3**: `anothertricky.py`
- **Java**: `AnotherTricky.java`. Note: The `AnotherTricky` class must contain your `main` method. Additional classes *must* be declared as private classes in the `AnotherTricky.java` file, **not** in additional `.java` files.
- **C++**: `AnotherTricky.cpp`

Your solution will read input from a file specified as a command-line parameter. You can assume that we will run your code as follows (where `FILENAME` is the name of a file containing the input for the problem):

- Python 3: `python3 anothertricky.py FILENAME`
- Java: `java AnotherTricky FILENAME`
- C++: `./AnotherTricky FILENAME` (assuming your file is compiled into an `AnotherTricky` binary)

Your code must print the output to standard output following the exact specification described below. Printing any other output (e.g., debug messages, log messages, etc.) to standard output will cause the tests to fail. That said, you are allowed to print anything to standard error (the tests only look at standard output, and ignore standard error).

As a reminder, your solution to this problem must be implemented using an object-oriented approach with multiple classes. Any other solution will receive zero points, even if the testing system deems it to be correct.

Input

The input begins with a line containing two integers, P and N, as defined earlier, separated by a single space. You can assume that $2 \leq P \leq N \leq 40$.

The input is followed by N lines, each specifying a card in the deck. Each card is specified by a single character (`A`, `B`, `C`, or `D`) followed by a space and followed by an integer between 1 and 10. You can assume there are no repeated cards in the input. In the game, cards are extracted from the deck in the order specified in the input.

Output

The output is composed of four lines. Each line contains a single character representing the trump suit used in the game (`A`, `B`, `C`, or `D`), followed by an integer representing the number of the winning player and an integer with the number of points they earned, each separated by a single space. The suits must appear in order (i.e., `A` in the first line, `B` in the second line, etc.)

Sample Input/Output

Test #1

Input

```
4 4
D 4
C 8
D 10
```

```
A 3
```

Output

```
A 4 14
B 3 14
C 2 14
D 3 14
```

Test #2

Input

```
4 8
D 4
C 8
D 10
A 3
D 7
A 4
D 6
B 5
```

Output

```
A 4 14
B 3 14
C 2 19
D 3 14
```