Your task is to develop an algorithm for comparing the strength of Texas Hold'em Hands. A value of a Texas Hold'em Hand is the best possible value out of all possible subsets of 5 cards from the 7 cards which are formed by 5 board cards and 2 hand cards.

The details of Texas Hold'em hand values are available on Wikipedia: https://en.wikipedia.org/wiki/Texas_hold_%27em#The_showdown, in particular, from weakest to strongest:

- High card - the "fallback" in case no other hand value rule applies
- Pair - two cards of the same value
- Two pairs - Two times two cards with the same value
- Three of a kind - Three cards with the same value
- Straight - Sequence of 5 cards in increasing value (Ace can precede 2 and follow up King)
- Flush - 5 cards of the same suit
- Full House - Combination of three of a kind and a pair
- Four of a kind - Four cards of the same value
- Straight Flush - straight of the same suit

In case of ties the ranks of the cards forming the combinations decide the highest value. In case of further ties, the ranks of remaining cards (in the order of the rank) decide the highest value. All suits are considered equal in strength.

When comparing Full Houses, the three of a kind rank comparison is more important than the pair rank comparison, for example, QQQ88 > 999KK, KKK77 > QQQJJ and KKK77 > KKK66.

When comparing Straights, the A2345 straight is the weakest one and the TJQKA one the strongest one, for example, 23456 > A2345 and TJQKA > 9TJQK.

If any of these rules are not clear, please consult the standard Texas Hold'em rules on Wikipedia.

The input is to be read from the standard input in the form of:

```
<5 board cards> <hand 1> <hand 2> <...> <hand N>
```

... where:

- <5 board cards> is a 10 character string where each 2 characters encode a card
- <hand X> is a 4 character string where each 2 characters encode a card, with 2 cards per hand
- <card> is a 2 character string with the first character representing the rank (one of "A", "K", "Q", "J", "T", "9", "8", "7", "6", "5", "4", "3", "2") and the second character representing the suit (one of "h", "d", "c", "s") .

The output is to be written to standard output using the format:

```
<hand block 1> <hand block 2> <...> <hand block n>
```

... where:

- <hand block 1> is the hand block with the weakest value

- <hand block 2> is the hand block with the second weakest value
- ... and so forth.
- <hand block n> is the hand block with the strongest value
- Each hand block consists of one or multiple hands (each represented by 4 character string with 2 characters to encode a card, with 2 cards per hand) with equal strength
- In case there are multiple hands with the same value on the same board they should be ordered alphabetically and separated by "=" signs
- The order of the cards in each hand should remain the same as in the input, e.g., don't reorder `2h3s` into `3s2h`.

For example:

```
Input:
4cKs4h8s7s Ad4s Ac4d As9s KhKd 5d6d
2h3h4h5d8d KdKs 9hJh

Output:
Ac4d=Ad4s 5d6d As9s KhKd
KdKs 9hJh
```

The application should read from standard input (stdin) until EOF is reached and write to standard output (stdout) streams. It will be tested using 25,000 of test cases using input and output redirection. One input line should correspond to exactly one output line.

Please do NOT output extra text or expect extra input besides the lines with test cases.

The homework can be written **in any programming language** which can be run on Ubuntu Linux and should include a "ReadMe.md" file with documentation on how to compile and run it, as well as install any prerequisites. The documentation should be in English.

Please make the application easy to build and test use Maven with Java, SBT with Scala, etc.

For extra credit you should:

- Implement the algorithm using Scala, and/or
- Implement a command line switch `--omaha` which uses 8 character strings representing 4 cards for each hand and evaluating hand values according to https://en.wikipedia.org/wiki/Omaha_hold_%27em rules,
- In case the input line is invalid, output a clear & easy to understand error message.

If you find this task too challenging, partial credit will be given for partial solutions. In this case, please make sure "ReadMe.md" clearly documents all known limitations of your solution.

A complete solution in a non-Scala language will be preferred to an incomplete solution in Scala.

Yielding correct results is more important than the algorithmic complexity of the solution.

Please publish the solution in a private GitHub repository and give user evo-home-task (e-mail dev-home-task@evolutiongaming.com) access to the repository.