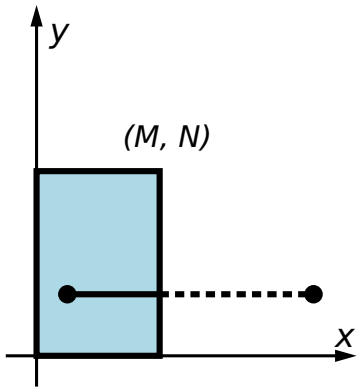
Rasteryzacja Odcinka

Odcinek w grafice komputerowej jest kodowany jako para wieszchołków. Zagadnienie rasteryzacji polega na najlepszym zobrazowaniu odcinka na ekranie. Napisz program, rasteryzujący poziome i pionowe odcinki.



Ekran jest reprezezentowany jako prostokątna siatka pikseli o całkowitych współrzędych. Współrzędne lewego dolnego rogu ekranu $(0, 0)$. Współrzędne prawego górnego rogu — (M, N). Wierzchołki odcinka mogą wychodzić za granice ekranu. Na rysunku obszar ekranu jest przyciemniony. Przerywana część odcinka wychodzi poza ekran i jest niewidoczna.

Wejście

W pierwszej linijce wejścia podana jest liczba testów K, a w następnych poszczególne testy. Każdy test składa się z dwóch linii. W pierwszej podane są dwie liczby całkowite, M i N, $(0 \leqslant M, N < 101)$ określające rozmiar ekranu. W drugiej linii podane są współrzędne wierzchołków odcinka (x_0, y_0) (x_1, y_1). Odcinek zawsze jest albo poziomy $(y_0 = y_1)$ albo pionowy $(x_0 = x_1)$. Wszystkie liczby są oddzielone spacją, jak w przykładzie.

Wyjście

Dla każdego testu należy wyświetlić ekran z rasteryzowanym odcinkiem. Piksele, w których nie ma odcinka, trzeba zakodować zerem, piksele odcinka — jedynką.

Przykład

Wejście

```
2
4 6
1 2 10 2
6 4
2 1 2 10
```

Wyjście

```
00000
00000
00000
00000
01111
00000
00000
0010000
0010000
0010000
0010000
0000000
```

Ułamki Egipskie

Ułamek nazywa się egipskim, jeżeli ma on jedność w liczniku.

Napisz program, który oblicza rozłożenie liczby wymiernej w sumę ułamków egipskich, poczynając od największego możliwego dla danej liczby ułamka. Pozostałe wyrazy sumy są formowane w sposób analogiczny do pierwszego. Na przykład:

$$15/16 = 1/2 + 1/3 + 1/10 + 1/240.$$

Wejście

Dane wejściowe rozpoczynają się od wiersza zawierającego jedną liczbę całkowitą dodatnią, oznaczająca liczbę zestawów danych. Każdy zestaw składa się z pary liczb całkowitych m i n ($0 < m < n < 32001$), prezdsawiających ułamek m/n.

Wyjście

Dla każdego zestawu wejściowego wydrukuj w jednym wierszu rozłożenie liczby m/n w sumę ułamków egipskich, sformatowane jak w przykładzie. Zwróć uwagę na odstępy dookoła znków „+” i „=”.

Przykład

Wejście

```
1
15 16
```

Wyjście

```
15/16 = 1/2 + 1/3 + 1/10 + 1/240
```

Sprawdź Hasło

Napisz program który sprawdzi poprawność podanych haseł. Poprawne hasła spełniają następujące warunki
— długość 6–20 znaków,
— brak powtarzających się sekwencji 2 lub 3 znakowych (np. nienie),
— przynajmniej jedna duża litera, jedna mała, jeden znak specjalny oraz jedna cyfra.

Wejście

Dane wejściowe rozpoczynają się od wiersza zawierającego jedną liczbę całkowitą dodatnią, oznaczająca liczbę zestawów danych. Każdy zestaw składa się z ciągu znaków reprezentujących hasło.

Wyjście

Dla każdego zestawu wejściowego wydrukuj w jednym wierszu odpowiedź "YES" dla poprawnego hasła i "NO" dla błędnego.

Przykład

Wejście

```
8
PASS
P@ss
P@ss123123
Alabama@#1
nonoP@ssw0rd
My2P@SSword
abcddcba123#
MyP@sswordIsReallyLong@RosesAreBlue
```

Wyjście

```
NO
NO
NO
YES
NO
YES
NO
NO
```

Sekretne Kody

Służby wywiadu Bajtocji przechwyciły ściśle tajne kody uruchomieniowe do rakiet dev/null wrogiego królestwa Bitocji. Pomóż agentom kontrwywiadu złamać kod.

Wiadomo, że kod ukryty jest pośród słów publikowanych w lokalnej prasie. Nie wszystkie znaki są ważne. Klucz stanowi N liter wraz z ich liczbowymi wagami.

Słowo kodowe dla jednego słowa stanowi sumę wag liczbowych dla liter wykrytych w kodzie. Agenci będą łamać kod metodą bruteforce.

Wejście

W pierwszej linii podana jest liczba z długością kodu — N, oraz K — ilość słów, kolejne N linii to kod, gdzie pierwsza stoi litera kodu a po spacji waga liczbowa, pozostałe K linii to słowa, dla których ma być wyliczona wartość kodu. Ograniczenia: w kodzie mogą znaleźć się tylko litery alfabetu łacińskiego. Małe i duże litery mogą mieć inną wagę. Słowo nie jest dłuższe niż 1000 znaków, $K < 10000$.

Wyjście

W pliku wyjściowym mają się znaleźć wartości kodu dla słów, po jednym w linijce.

Przykład

Wejście

```
4 3
A 56
a 21
B 42
c 11
Alpy
Baca
Owca
```

Wyjście

```
56
95
32
```

MinMax

Danych jest $N < 1024$ rzędów osób. Twoim zadaniem jest znaleźć najniższą osobę spośród najwyższych osób z każdego rzędu oraz analogicznie najwyższą osobę spośród najniższych z każdego rzędu.

Wejście

Każda linia wejściowa składa się z maksymalnie 100 liczb odzielonych spacją reprezentujących wzrost osób w danym rzędzie. Dane wejściowe kończy pusty wiersz.

Wyjście

Na wyjściu należy podać dwie liczby w osobnych wierszach. Pierwsza to wzrost najniższej osoby z najwyższych, druga — najwyższej spośród najniższych.

Przykład

Wejście

```
170 190 168 155
198 205 183 174 165
167 177 201
```

Wyjście

```
190
167
```

Zapytania

Dany jest obiekt zserializowany do formatu JSON. Twoim zadaniem jest przetworzyć wszystkie zapytania i zwrócić wartości atrybutów tego obiektu.

Wejście

Dane wejściowe składają się z dwóch części oddzielonych pustym wierszem.

Pierwsza część to obiekt w uproszczonym formacie JSON. Każdy obiekt opisany jest między klamrami `{ ... }` i posiada oddzielone przecinkiem atrybuty w formacie `"nazwa" : wartość`. Wartość może być napisem, kolejnym obiektem `{ ... }` bądź tablicą `[ ... ]` wartości/obiektów (indeksowana od 0, elementy oddzielone przecinkiem). Białe znaki nie mają żadnego znaczenia i zostały dodane tylko na potrzeby formatowania. Wartości napisowe zawsze rozpoczyna i kończy cudzysłów, pomiędzy którymi mogą być dowolne inne znaki.

Druga część — to ścieżki, wskazujące do konkretnych wartości obiektu (każda ścieżka w osobnej linii). Na potrzeby zadania możesz założyć, że poprawna ścieżka zawsze wskazuje wartość napisową. Dane wejściowe kończy pusty wiersz.

Wyjście

Dla każdej ścieżki wejściowej należy wypisać powiązaną z nią wartość opuszczając znaki cudzysłowu `""`. Jeśli ścieżka jest niepoprawna i wskazuje na nieistniejący atrybut należy ją pominąć.

Przykład

Wejście

```
{
 "firstName": "John",
 "lastName": "Smith",
 "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
 },
 "phoneNumbers": [
    {
        "type": "home",
        "number": "212 555-1234"
    },
    {
        "type": "fax",
        "number": "646 555-4567"
    }
 ]
}

firstName
address.city
address.country
phoneNumbers[0].type
phoneNumbers[1].number
```

Wyjście

```
John
New York
home
646 555-4567
```