Konnichiwa OIE 2018

Este año la olimpiada internacional de informática se va a celebrar en Japón. La moneda japonesa es el Yen Japonés, actualmente un euro equivale, aproximadamente, a 130 Yenes japoneses.

Haced un programa que pase de euros a yenes y de yenes a euros. En caso que el resultado no sea entero y tenga decimales, estos no se tendrán en cuenta.

Input Format

La entrada empieza con una línea con un entero n.

Cada una de las siguientes n líneas contiene un caso, cada caso consiste en un número entero m_i seguido de una letra l_i que indica el tipo de moneda: ***E*** para euros y ***Y*** para yenes.

Constraints

$1 \leq n \leq 100000$

$0 \leq m_i \leq 1000000$

25 Puntos: Todos los casos son euros.

25 Puntos: Todos los casos son yenes.

50 Puntos: Todos tipo de casos.

Output Format

Para cada caso escribid una línea con un entero con la conversión a euros o yenes, dependiendo del tipo de moneda original, seguido de la letra del nuevo tipo de moneda: ***E*** para euros y ***Y*** para yenes.

Sample Input 0

```
3
1 E
130 Y
0 Y
```

Sample Output 0

```
130 Y
1 E
0 E
```

Sample Input 1

```
2
2 E
150 Y
```

Sample Output 1

```
260 Y
1 E
```

Solución:

```cpp
#include <iostream>

using namespace std;

int main() {
  int n;
  cin >> n;
  while (n--) {
    int k;
    char a;
    cin >> k >> a;
    if (a == 'E') {
      cout << 130*k << " Y\n";
    } else {
      k /= 130;
      cout << k << " E\n";
    }
  }
}
```

Haciendo sushi

Takeshi Kitano está preparando sushi para sus n amigos. Sus amigos tienen gustos muy distintos y a cada uno le gusta un tipo de sushi diferente. Cada tipo de sushi usa g_i gramos de pescado. Para que haya más variedad Takeshi quiere comprar n tipos de pescado diferentes, uno para cada tipo de sushi. Cada pescado cuesta p_i yenes el gramo.

Dados los g_i y los p_i, qual es el mínimo precio que tiene que pagar Takeshi Kitano para preparar sushi para sus amigos?

Cada tipo de sushi solo puede usar un tipo de pescado y se usan todos los tipos de pescado, uno para cada tipo de sushi.

Input Format

La entrada empieza con un entero t, seguido de t casos.

Cada caso contiene una línea con un entero n. La segunda línea contiene los n enteros g_i. La tercera línea contiene los n enteros p_i.

Constraints

$1 \leq t \leq 100$

$1 \leq n \leq 100000$

$n_{max} * t \leq 2500000$

Donde n_{max} es el mayor valor de n en el fichero de entrada.

$1 \leq g_i, p_i \leq 1000000$

20 Puntos: $1 \leq n \leq 10$

20 Puntos: $1 \leq n \leq 100$

20 Puntos: $1 \leq n \leq 1000$

20 Puntos: $1 \leq n \leq 10000$

20 Puntos: $1 \leq n \leq 100000$

Output Format

Para cada caso escribid una línea con el precio mínimo que tiene que pagar Takeshi Kitano.

Atención: el resultado puede ser muy grande y no caber en un *int*, por eso hay que usar *long long int* en C++ o su equivalente en otros lenguajes.

Sample Input 0

```
1
5
3 2 4 2 4
3 1 2 2 3
```

Sample Output 0

```
30
```

Sample Input 1

```
2
4
7 8 6 4
6 7 3 10
4
3 8 1 10
4 7 1 7
```

Sample Output 1

```
148
70
```

Solución:

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

using namespace std;

typedef long long ll;

int main() {
  int t;
  cin >> t;
  while (t--) {
    int n;
    cin >> n;
    vector<ll> G(n), P(n);
    for (int i = 0; i < n; ++i) cin >> G[i];
    for (int i = 0; i < n; ++i) cin >> P[i];
    sort (G.begin(), G.end());
    sort(P.begin(), P.end());
    ll res = 0;
    for (int i = 0; i < n; ++i)
      res += G[i]*P[n-1-i];
    cout << res << endl;
  }
}
```

Vuelos a Japón

El equipo organizador de la OIE quiere comprar ya los vuelos a Japón para ir a la olimpiada internacional. El problema es que desde Barcelona no siempre hay vuelos directos a Tokyo.

Al equipo español no le importa el tiempo que tarden en llegar a Japón, solo cuanto va a costar. Cada vuelo sale de una ciudad s_i y llega a una ciudad l_i con un coste c_i. Solo se puede coger un avión que sale de una ciudad x si anteriormente se ha llegado a esta ciudad con otro vuelo o es la ciudad inicial.

Si Barcelona es la ciudad 0 i Tokyo la ciudad 1, cual es el mínimo precio para ir de Barcelona a Japón en avión?

Input Format

La entrada consiste en un entero t seguido de t casos.

Cada caso empieza con un entero n, el número de ciudades y un entero m, el número de vuelos. Las siguientes m líneas contienen la información de cada vuelo: s_i, l_i, c_i.

Constraints

$0 \leq s_i, l_i < n$

$0 < c_i \leq 1000$

$2 \leq n \leq 100000$

$0 \leq m \leq 100000$

$1 \leq t \leq 30$

33 Puntos: $n, m \leq 1000$

33 Puntos: $n, m \leq 10000$

34 Puntos: $n, m \leq 100000$

Output Format

Escribid una línea con el mínimo coste para ir desde la ciudad 0 a la ciudad 1.

En caso que no exista una combinación de vuelos para llegar de 0 a 1, escribid -1.

Sample Input 0

```
1
3 4
0 1 2
1 2 3
0 1 5
1 2 5
```

Sample Output 0

```
2
```

Sample Input 1

```
3
2 3
0 1 2
1 0 4
1 0 4
2 3
1 0 1
0 1 4
1 0 4
2 3
0 1 2
0 1 4
1 0 4
```

Sample Output 1

```
2
4
2
```

Sample Input 2

```
1
3 2
0 2 3
1 2 4
```

Sample Output 2

```
-1
```

Solución:

```
#include <iostream>
#include <cstring>
#include <vector>
#include <queue>

using namespace std;

typedef pair<int, int> pi;
typedef vector<pi> vpi;
typedef vector<vpi> vvpi;
typedef vector<int> vi;

const int N = 100010;
vpi G[N];
int D[N];
```

```
int main() {
  int t;
  cin >> t;
  while (t--) {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; ++i) G[i].clear();
    memset(D, -1, sizeof(D));
    for (int i = 0; i < m; ++i) {
      int x, y, l;
      cin >> x >> y >> l;
      G[x].push_back(pi(y, l));
    }
    priority_queue<pi> Q;
    Q.push(pi(0,0));
    D[0] = 0;
    while (!Q.empty()) {
      int x = Q.top().second;
      int l = -Q.top().first;
      if (x == 1) break;
      Q.pop();
      if (D[x] != l) continue;
      for (int i = 0; i < (int)G[x].size(); ++i) {
        int y = G[x][i].first;
        int c = G[x][i].second;
        if (D[y] == -1 or D[y] > D[x] + c) {
          D[y] = D[x] + c;
          Q.push(pi(-D[y], y));
        }
      }
    }
    cout << D[1] << endl;
  }
}
```