Friend

Construimos una red social de n personas numeradas de 0, ... , $n-1$. Algunos pares de personas en la red serán amigos. Si la persona x es amigo de la persona y, entonces la persona y también es amigo de la persona x.

Las personas son añadidas a la red en n etapas, las cuales son numeradas de 0 a $n-1$. Es decir la persona i es añadida en la etapa i. En la etapa 0, la persona 0 es añadida como la unica persona en la red. En cada una de las siguientes $n-1$ etapas, una persona es añadida a la red por un (host), quien puede ser cualquier persona que ya esta en la red. En la etapa i ($0 < i < n$), el host puede añadir a la persona i dentro la red siguiendo uno de los siguientes 3 protocolos.

- *IamYourFriend* hace a la persona i amigo de solo el host.
- *MyFriendsAreYourFriends* hace a la persona i amigo de *cada uno* de los amigos del host. Nota que este protocolo *no* hace que la persona i sea un amigo del host.
- *WeAreYourFriends* hace a la persona i amigo del host, y tambien amigo de *cada uno* de los amigos del host.

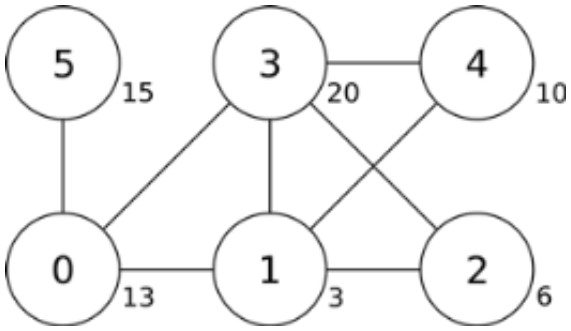
Despues de construir la red nos gustaria escoger una *muestra* para una encuesta, esto quiere decir, escoger un grupo de personas de la red. Ya que los amigos normalmente tienen intereses iguales, la muestra no debe incluir a personas que son amigos. Cada persona tiene una encuesta de *confidencia* expresada como un entero positivo, y nos gustaria encontrar una muestra con el máximo total de *confidencia*.

Example

stage	host	protocol	friend relations added
1	0	IamYourFriend	(1, 0)
2	0	MyFriendsAreYourFriends	(2, 1)
3	1	WeAreYourFriends	(3, 1), (3, 0), (3, 2)
4	2	MyFriendsAreYourFriends	(4, 1), (4, 3)
5	0	IamYourFriend	(5, 0)

Inicialmente la red contiene solo a la persona 0. El host de la etapa 1 (en este caso la persona 0) invita a la persona 1 con el protocolo IamYourFriend, por lo tanto ellos son amigos. El host de la etapa 2 (persona 0 nuevamente) invita a la persona 2 con MyFriendsAreYourFriends, lo cual hace a la persona 1 (único amigo del host) el unico amigo de la persona 2. El host de la etapa 3 (persona 1) añade a la persona 3 con WeAreYourFriends, el cual hace a la persona 3 un amigo de la persona 1 (el host) y las personas 0 y 2 (los amigos del host). La etapa 4 y 5 son tambien mostradas en la tabla de arriba. la red final es mostrada en la imagen a continuación, los numeros dentro los circulos muestran las etiquetas de las personas, y los números al lado de los circulos muestran las encuestas de

confidencia. La muestra consistente de las personas 3 y 5 tienen un total de encuesta de confidencias igual a 20 + 15 = 35, el cual es el máximo posible total de confidencia.



Task

Dada la descripción de cada etapa y el valor de confidencia de cada persona, encontrar una muestra con el máximo total de confidencia. Tu solo necesitas implementar la función `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n`: el número de personas.
  - `confidence`: array de longitud n; `confidence[i]` que contiene el valor de confidencia de la persona i.
  - `host`: array de longitud n; `host[i]` que contiene al host de la etapa i.
  - `protocol`: array de longitud n; `protocol[i]` que contiene el código del protocolo usado en la etapa i ($0 < i < n$): 0 for IamYourFriend, 1 for MyFriendsAreYourFriends, and 2 for WeAreYourFriends.
  - Ya que no hay host en la etapa 0, `host[0]` y `protocol[0]` son indefinidos y no deberían ser procesados por tu programa.
  - La función debería retornar el máximo posible total de confidencia.

Subtasks

Some subtasks use only a subset of protocols, as shown in the following table.

subtask	points	n	confidence	protocols used
1	11	$2 \leq n \leq 10$	$1 \leq \text{confidence} \leq 1,000,000$	All three protocols
2	8	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Only MyFriendsAreYourFriends
3	8	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Only WeAreYourFriends
4	19	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Only IamYourFriend
5	23	$2 \leq n \leq 1,000$	All confidence values are 1	Both MyFriendsAreYourFriends and IamYourFriend
6	31	$2 \leq n \leq 100,000$	$1 \leq \text{confidence} \leq 10,000$	All three protocols

Implementation details

You have to submit exactly one file, called `friend.c`, `friend.cpp` or `friend.pas`. This file should implement the subprogram described above, using the following signatures. You also need to include a header file `friend.h` for C/C++ implementation.

C/C++ program

```
int findSample(int n, int confidence[], int host[], int protocol[]);
```

Pascal programs

```
function findSample(n: longint, confidence: array of longint, host: array
of longint; protocol: array of longint): longint;
```

Sample grader

The sample grader reads the input in the following format:

- line 1: `n`
- line 2: `confidence[0]`, ..., `confidence[n-1]`
- line 3: `host[1]`, `protocol[1]`, `host[2]`, `protocol[2]`, ..., `host[n-1]`, `protocol[n-1]`

The sample grader will print the return value of `findSample`.