Friend

Construiremos una red social de n personas numeradas 0,...,$n-1$. En la red, algunas personas serán amigos entre ellos. Si una persona x se hace amiga de una persona y, entonces la persona y también se hace amiga de la persona x.

Las personas son añadidas a la red en n etapas, las que también están numeradas de 0 a $n-1$. La persona i es añadida en la etapa i. En la etapa 0, la persona 0 es añadida como la única persona en la red. En cada una de las siguientes $n-1$ etapas, una persona es añadida a la red por un *host*, quién puede ser cualquier persona que ya se encuentre en la red. En la etapa i (para $0 < i < n$), el *host* de esa etapa puede añadir a la persona i a la red a través de uno de los siguientes protocolos:

- *IAmYourFriend* hace que la persona i sea amiga sólo del *host*
- *MyFriendsAreYourFriends* hace que la persona i sea amiga de cada una de las personas que son amigas del *host* en ese momento. Notar que este protocolo *no* hace que la persona i sea amiga del *host*.
- *WeAreYourFriends* hace que la persona i sea amiga del *host* y también de todos las personas que son amigas del *host* en ese momento.

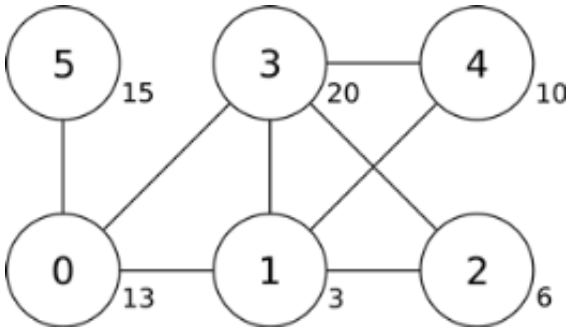
Después de construir la red nos gustaría elegir una *muestra* para una encuesta, esto es, escoger un grupo de personas en la red. Dado que los amigos usualmente tienen intereses similares, la muestra no debe incluir ningún par de personas que son amigos entre ellos. Cada persona tiene además un indicador de *confianza* para la encuesta, expresada como un entero positivo, y nos gustaría encontrar una muestra que maximice la confianza total.

Ejemplo

etapa	host	protocolo	relación de amistad agregada
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 sólo a la persona 0. El *host* de la etapa 1 (persona 0) invita a la persona 1 a través del protocolo *IAmYourFriend*, y por lo tanto ellos se transforman en amigos. El *host* de la etapa 2 (nuevamente la persona 0) invita a la persona 2 usando *MyFriendsAreYourFriends*, lo que hace que la persona 1 (el único amigo del *host*) sea el único amigo de la persona 2. El *host* de la etapa 3 (persona 1) añade a la persona 3 a través de *WeAreYourFriends*, lo que convierte a la persona 3 en amigo de la persona 1 (el *host*) y las persona 0 y 2 (los amigos del *host*). Las etapas 4 y 5 también son mostradas en la tabla de arriba. La red final está mostrada en la figura de abajo, en la cual los

números al interior de los círculos representan las etiquetas de las personas, y los números contiguos a los círculos representan su *confianza*. En la muestra compuesta por las personas 3 y 5 tienen una *confianza* total de 20 + 15 = 35, que es el número máximo de confianza total posible.



Tarea

Dada la descripción de cada etapa y el valor de la confianza asociado a cada persona, debes encontrar la muestra con la máxima confianza total. Sólo debes implementar la función `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n`: el número de personas.
  - `confidence`: arreglo de tamaño n; `confidence[i]` contiene la confianza asociada a la persona i.
  - `host`: arreglo de tamaño n; `host[i]` contiene el *host* de la etapa i.
  - `protocol`: arreglo de tamaño n; `protocol[i]` contiene el código del protocolo usado en la etapa i ($0 < i < n$): 0 para *IAmYourFriend*, 1 para *MyFriendsAreYourFriends*, y 2 para *WeAreYourFriends*.
- Dado que no hay *host* en la etapa 0, `host[0]` y `protocol[0]` están indefinidos y tu programa no debe acceder a ellos.
- La función debe retornar el número máximo de confianza total posible para una muestra.

Subtareas

Algunas subtareas usan solo un subconjunto de protocolos, como se muestra en la siguiente tabla.

subtarea	puntos	n	confianza	protocolos que puede ser usados
1	11	$2 \leq n \leq 10$	$1 \leq \text{confianza} \leq 1,000,000$	Los tres protocolos
2	8	$2 \leq n \leq 1,000$	$1 \leq \text{confianza} \leq 1,000,000$	Sólo *MyFriendsAreYourFriends*
3	8	$2 \leq n \leq 1,000$	$1 \leq \text{confianza} \leq 1,000,000$	Sólo *WeAreYourFriends*
4	19	$2 \leq n \leq 1,000$	$1 \leq \text{confianza} \leq 1,000,000$	Sólo *IAmYourFriend*
5	23	$2 \leq n \leq 1,000$	Todas las confianzas son 1	*MyFriendsAreYourFriends* y *IAmYourFriend*
6	31	$2 \leq n \leq 100,000$	$1 \leq \text{cconfianza} \leq 10,000$	Los tres protocolos

Detalles de implementación

Debes enviar exactamente un archivo, llamado `friend.c`, `friend.cpp` o `friend.pas`. Este archivo debe contener la implementación del subprograma descrito anteriormente y usando la firma descrita a continuación. Para programas en C/C++ debes incluir adicionalmente el archivo `friend.h`.

C/C++

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

Pascal

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

Grader de ejemplo

El *grader* de ejemplo lee el input en el siguiente formato:

- 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]`

El *grader* de ejemplo imprimirá el valor retornado por `findSample`.