Friend

Vi bygger ett socialt nätverk med n personer numrerade 0, ... , $n-1$. Vissa par av personer i nätverket kommer att bli vänner. Om en person x blir vän med en person y, så blir även y vän med x.

De n personerna läggs till i nätverket i n faser, som också är numrerade från 0 till $n-1$. Person i läggs till under fas i. Under fas 0, så läggs person 0 till som enda person i nätverket. Under kommande $n-1$ faser så läggs en person till av en *värd*, som kan vara vilken person som helst som redan finns i nätverket. Under fas i $(0 < i < n)$ så kan värden för nuvarande fas lägga till person i till nätverket enligt ett av tre följande protokoll:

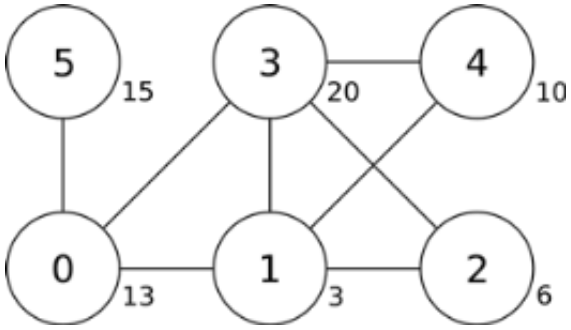
- *IAmYourFriend* gör person i till vän med endast värden.
- *MyFriendsAreYourFriends* gör person i till vän med *alla dåvarande värdens vänner*. Notera att detta protokoll *inte* gör person i till vän med värden.
- *WeAreYourFriends* gör person i till vän med värden, och även till vän med *alla dåvarande* värdens vänner.

Efter att vi byggt nätverket så vill vi välja ett *urval* av personerna för en undersökning, d.v.s. välja ut en delmängd av personerna i nätverket. Eftersom vänner ofta har liknande intressen så får inte urvalet innehålla två personer som är vänner med varandra. Varje person har en *pålitlighet*, uttryckt som ett positivt heltal, och vi skulle vilja hitta ett urval med maximal total pålitlighet.

Exempel

fas	värd	protokoll	tillagda vänskapsrelationer
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)

Till en början innehåller nätverket endast person 0. Värden för fas 1 (person 0) bjuder in person 1 genom "IAmYourFriend"-protokollet och de blir vänner. Värden för fas 2 (person 0 igen) bjuder nu in person 2 genom "MyFriendsAreYourFriends"-protokollet, vilket gör person 1 (den enda vännen till värden) till den enda vännen till person 2. Värden för fas 3 (person 1) lägger till person 3 genom "WeAreYourFriends"-protokollet, vilket gör person 3 till vän med person 1 (värden) och person 0 och 2 (värdens vänner). Faserna 4 och 5 visas även i tabellen ovan. Det slutgiltiga nätverket visas i följande figur, i vilken siffrorna i cirklarna indikerar personernas siffra, och talen bredvid cirklarna indikerar deras pålitlighet. Urvalet som innehåller personerna 3 och 5 har total pålitlighet 20 + 15 = 35, vilken är den maximala möjliga pålitligheten.



Uppgift

Givet beskrivningen av varje fas och pålitligheten för varje person, hitta ett urval som ger största möjliga pålitlighet. Du behöver bara implementera funktionen `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n`: antalet personer.
  - `confidence`: array av längd n; `confidence[i]` ger pålitligheten för person i.
  - `host`: array av längd n; `host[i]` ger värden för fas i.
  - `protocol`: array av längd n; `protocol[i]` anger protokollet som används under fas i ($0 < i < n$): 0 för "IAmYourFriend", 1 för "MyFriendsAreYourFriends", och 2 för "WeAreYourFriends".
  - Eftersom det inte finns någon värd under fas 0 så är `host[0]` och `protocol[0]` odefinierade och ska inte läsas av ditt program.
  - Funktionen ska returnera den maximala möjliga pålitligheten för ett urval.

Deluppgifter

Vissa deluppgifter använder bara en delmängd av protokollen, som visas i följande tabell.

deluppgift	poäng	n	pålitlighet	använda protokoll
1	11	$2 \leq n \leq 10$	$1 \leq$ pålitlighet $\leq 1,000,000$	Alla tre protokoll
2	8	$2 \leq n \leq 1,000$	$1 \leq$ pålitlighet $\leq 1,000,000$	Endast "MyFriendsAreYourFriends"
3	8	$2 \leq n \leq 1,000$	$1 \leq$ pålitlighet $\leq 1,000,000$	Endast "WeAreYourFriends"
4	19	$2 \leq n \leq 1,000$	$1 \leq$ pålitlighet $\leq 1,000,000$	Endast "IAmYourFriend"
5	23	$2 \leq n \leq 1,000$	Alla pålitligheter har värde 1	Både "MyFriendsAreYourFriends" och "IAmYourFriend"
6	31	$2 \leq n \leq 100,000$	$1 \leq$ pålitlighet $\leq 10,000$	Alla tre protokoll

Implementationsdetaljer

Du ska skicka in exakt en fil med namn `friend.c`, `friend.cpp` eller `friend.pas`. Denna fil ska implementera subprogrammet som beskrivits ovan, med signaturer som följer nedan. Du ska även inkludera header-filen `friend.h` om du använder C/C++.

C/C++-program

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

Pascal-program

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

Exempelrättare (sample grader)

Exempelrättaren läser indatan enligt följande format:

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

Exempelrättaren kommer att printa returvärdet från `findSample`.