Friend

Vi bygger et sosialt nettverk fra n personer nummerert $0, ..., n-1$. Noen par av personer i nettverket vil være venner. Dersom person x blir venn med person y så blir automatisk person y venn med person x.

Folk legges til i nettverket i n trinn som også er nummerert fra 0 til $n-1$. Person i legges til i trinn i. I trinn 0 så blir person 0 lagt til som den eneste personen i nettverket. I hver av de neste $n-1$ trinnene så blir en person lagt til i netverket av en *vert* (*host* på engelsk), som kan være en hvilken som helst person som allerede befinner seg i nettverket. I trinn i ($0 < i < n$), så kan verten for det trinnet legge til person i i nettverket med en av følgende tre protokoller:

- *JegErDinVenn* gjør person i kun venn med verten.
- *MineVennerErDineVenner* gjør person i til venn med *hver* av vertens nåværende venner. Merk at denne protokollen gjør *ikke* at person i blir en venn av verten.
- *ViErDineVenner* gjør person i til venn med verten og *hver* av vertens nåværende venner.

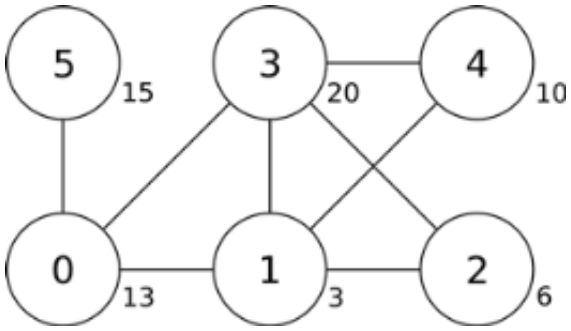
Etter at vi har bygd nettverket så ønsker vi å gjøre et *utvalg* for en undersøkelse. Det vil si at vi øsnker å velge en samling med personer fra nettverket. Siden venner ofte har lignende interesser så krever vi at utvalget ikke skal inneholde noen personer som er venner med hverandre. Hver person har en *undersøkelsestillit* (*confidence* på engelsk), uttrykt som et positivt heltall, og vi ønsker å finne det utvalget som har størst total tillit.

Eksempel

trinn	vert	protokol	vennerelasjoner lagt til
1	0	JegErDinVenn	(1, 0)
2	0	MineVennerErDineVenner	(2, 1)
3	1	ViErDineVenner	(3, 1), (3, 0), (3, 2)
4	2	MineVennerErDineVenner	(4, 1), (4, 3)
5	0	JegErDinVenn	(5, 0)

Til å begynne med så inneholder nettverket bare person 0.
Verten til trinn 1 (person 0) inviterer person 1 gjennom JegErDinVenn-protokollen så de blir venner.
Verten for trinn 2 (person 0 igjen) inviterer person 2 gjennom MineVennerErDineVenner-protokollen, noe som gjør at person 1 (den eneste vennen til verten) blir venn med person 2.
Verten for trinn 3 (person 1) legger til person 3 gjennom ViErDineVenner, noe som gjør at person 3 blir venn med person 1 (verten) og personer 0 og 2 (vennene til verten).
Trinn 4 og 5 er også beskrevet i tabellen over. Det ferdige nettverket vises i figuren under, hvor tallene i sirklene viser personenes tall-etikett og tallene ved siden av sirkene viser personens undersøkelsestillit. Utvalget bestående av person 3 og 5 har total undersøkelsestillit lik $20 + 15 = 35$,

noe som er den største mulige tilliten man kan oppnå.



Oppgave

Gitt beskrivelsen av hvert trinn og tillitsverdien til hver person, finn et utvalg med størst mulig total tillit. Du trenger bare å implementere funksjonen `findSample`.

- `findSample(n, confidence, host, protocol)`
  - `n`: antall personer.
  - `confidence`: et array med lengde n; `confidence[i]` gir tillitsverdien til person i.
  - `host`: et array med lengde n; `host[i]` gir verten for trinn i.
  - `protocol`: et array med lengde n; `protocol[i]` gir protokollen som blir brukt i trinn i ($0 < i < n$): 0 for JegErDinVenn, 1 for MineVennerErDineVenner, og 2 for ViErDineVenner.
  - Siden det ikke er noen vert i trinn 0 så er `host[0]` og `protocol[0]` udefinerte og bør ikke aksesseres av programmet ditt.
  - Funksjonen skal returnere den største mulige totale tillitsverdien til et utvalg.

Deloppgaver

Noen deloppgaver bruker kun noen av protokollene, vist i tabellen under.

deloppgave	poeng	n	tillit	protokoller brukt
1	11	$2 \leq n \leq 10$	$1 \leq \text{confidence} \leq 1,000,000$	Alle tre protokollene
2	8	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Bare MineVennerErDineVenner
3	8	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Bare ViErDineVenner
4	19	$2 \leq n \leq 1,000$	$1 \leq \text{confidence} \leq 1,000,000$	Bare JegErDinVenn
5	23	$2 \leq n \leq 1,000$	Alle tillitsverdiene er 1	Bare MineVennerErDineVenner og JegErDinVenn
6	31	$2 \leq n \leq 100,000$	$1 \leq \text{confidence} \leq 10,000$	Alle tre protokollene

Implementasjonsdetaljer

Du må levere inn nøaktig én fil, ved navn `friend.c`, `friend.cpp` eller `friend.pas`. Denne filen skal implementere funksjonen beskrevet ovenfor med de følgende signaturene. For C/C++-implementasjoner trenger du også å inkludere header filen `friend.h`.

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;
```

Eksempel-grader

Eksempel-graderen leser input i følgende format:

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

Eksempel-graderen skriver ut verdien returnert av `findSample`.