友達 (Friend)

0 から $n-1$ までの番号がついた n 人からなるソーシャルネットワークを作ろう．ネットワーク内のいくつかの 2 人組が友達となる．人 x が人 y の友達になるならば，人 y は人 x の友達にもなる．

n 人は，0 から $n-1$ までの番号がついた n 回の操作によってネットワークに加えられていく．人 i は操作 i で加えられる．操作 0 では人 0 が加わり，ネットワークのただ 1 人となる．続く $n-1$ 回の操作のそれぞれでは，既にネットワーク内にいる誰か 1 人が**ホスト** (host) となる．操作 i $(0 < i < n)$ では，以下の 3 種類の**方式** (protocol) のいずれかを用いてホストは人 i をネットワークに加える：

- IAmYourFriend： 人 i は，ホストのみと友達になる．
- MyFriendsAreYourFriends： 人 i は，その時点でのホストの友達**それぞれ**と友達になる．この方式では人 i はホストとは友達に**ならない**ことに注意せよ．
- WeAreYourFriends： 人 i は，ホストと友達になる．また，その時点でのホストの友達**それぞれ**とも友達になる．

ネットワークを作ったのち，あるアンケート調査の**サンプル** (sample) を選ぶことになった．ネットワークから何人かをサンプルとして選びたい．友達どうしは関心が似ている場合が多いため，サンプルには友達であるような 2 人組を含めてはならない．n 人のそれぞれには，調査の**信頼度** (confidence) と呼ばれる正の整数が定まっている．信頼度の合計が最大であるようなサンプルを求めたい．

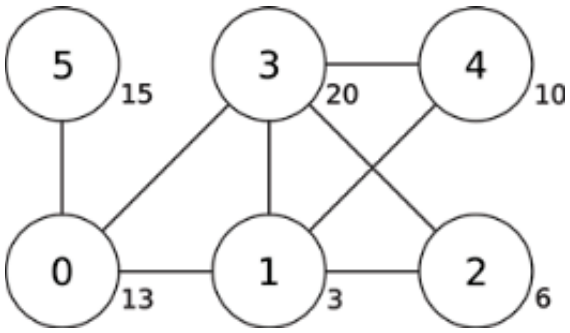
例 (Example)

操作	ホスト	方式	増えた友達関係
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)

はじめ，ネットワークは人 0 のみを含む．

1. 操作 1 のホスト (人 0) は人 1 を IAmYourFriend によって加える．人 0 と人 1 は友達となる．
2. 操作 2 のホスト (人 0) は人 2 を MyFriendsAreYourFriends によって加える．ホストの友達である人 1 のみが，人 2 と友達となる．
3. 操作 3 のホスト (人 1) は人 3 を WeAreYourFriends によって加える．人 1 (ホスト) および，人 0 と人 2 (ホストの友達) が，人 3 と友達になる．

操作 4 と操作 5 も同様である．最終的なネットワークが次の図で示されている (円の中の数は人の番号を，円の隣の数は信頼度を表す)．人 3 と人 5 からなるサンプルを考えると，信頼度の総和は $20 + 15 = 35$ となり，これが信頼度の総和の最大値である．



課題 (Task)

各操作の説明と各人の信頼度が与えられたとき，サンプルの信頼度の総和の最大値を求めよ．関数 `findSample` を実装せよ：

- `findSample(n, confidence, host, protocol)`
  - `n`： 人数.
  - `confidence`： サイズ n の配列であり，`confidence[i]` は人 i の信頼度を表す.
  - `host`： サイズ n の配列であり，`host[i]` は操作 i のホストの番号を表す $(0 < i < n)$.
  - `protocol`： サイズ n の配列であり，`protocol[i]` は操作 i で用いられる方式を表す $(0 < i < n)$. 0 は IAmYourFriend を，1 は MyFriendsAreYourFriends を，2 は WeAreYourFriends を表す.
  - 操作 0 にはホストがいないので，`host[0]` と `protocol[0]` は未定義であり，あなたのプログラムでアクセスしてはならない.
  - この関数は，サンプルの信頼度の総和の最大値を返すこと.

小課題 (Subtasks)

以下の表に示す通り，いくつかの小課題では，操作において方式のうちのいくつかだけが用いられる.

小課題	得点	n	信頼度 (confidence)	用いられうる方式 (protocol)
1	11	$2 \leqq n \leqq 10$	$1 \leqq$ 信頼度 $\leqq 1,000,000$	3 つすべて
2	8	$2 \leqq n \leqq 1,000$	$1 \leqq$ 信頼度 $\leqq 1,000,000$	MyFriendsAreYourFriends のみ
3	8	$2 \leqq n \leqq 1,000$	$1 \leqq$ 信頼度 $\leqq 1,000,000$	WeAreYourFriends のみ
4	19	$2 \leqq n \leqq 1,000$	$1 \leqq$ 信頼度 $\leqq 1,000,000$	IAmYourFriend のみ
5	23	$2 \leqq n \leqq 1,000$	信頼度はすべて 1 である	MyFriendsAreYourFriends と IAmYourFriend
6	31	$2 \leqq n \leqq 100,000$	$1 \leqq$ 信頼度 $\leqq 10,000$	3 つすべて

実装の詳細 (Implementation details)

1 つのファイルを提出せよ．提出するファイルの名前は `friend.c`, `friend.cpp`, `friend.pas` のいずれかである．このファイルには課題で指定されたサブプログラムを以下のシグネチャを用いて実装すること．C/C++ のプログラムにおいては，`friend.h` をインクルード (include) する必要がある．

C/C++ プログラム (C/C++ program)

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

Pascal プログラム (Pascal program)

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

採点プログラムのサンプル (Sample grader)

採点プログラムのサンプルは，以下のフォーマットで入力を読み込む：

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

採点プログラムのサンプルは，`findSample` の戻り値を出力する．