1. Bankomaty

Zadanie

W sali obsługi klientów banku kredytowego ustawiono 100 bankomatów ponumerowanych od 0 do 99. Każdy bankomat wykonuje tylko jedną operację: wypłaca albo przyjmuje ustaloną kwotę. Bankomat o numerze i wypłaca 2^i złotych, jeśli i jest parzyste, zaś przyjmuje 2^i złotych, jeśli i jest nieparzyste. Gdy klient zamierza pożyczyć ustaloną kwotę, trzeba zbadać, czy będzie mógł ją pobrać, korzystając co najwyżej raz z każdego z bankomatów i jeśli tak, wyznaczyć numery bankomatów, z których należy skorzystać. Trzeba również zbadać, czy będzie mógł ją zwrócić w podobny sposób i jeśli tak, wyznaczyć numery bankomatów, z których należy skorzystać w celu wykonania tej operacji.

Przykład:

Klient, który zamierza pożyczyć 7 złotych, pobiera najpierw 16 złotych w bankomacie nr 4 i 1 złoty w bankomacie nr 0, a następnie oddaje 8 złotych w bankomacie nr 3 i 2 złote w bankomacie nr 1. Żeby zwrócić pożyczoną kwotę 7 złotych, pobiera najpierw 1 złoty w bankomacie numer 0, a następnie oddaje 8 złotych w bankomacie nr 3.

Napisz program, który:

- wczytuje ze standardowego wejścia wysokość kwoty, jaką klient zamierza pożyczyć,
- sprawdza, czy klient będzie mógł pobrać ustaloną kwotę korzystając co najwyżej raz z każdego bankomatu i jeśli tak, wyznacza numery bankomatów, z których należy skorzystać, oraz czy będzie mógł ją zwrócić w podobny sposób i jeśli tak, wyznacza numery bankomatów, których należy użyć w tym celu.

Wejście

W pierwszym i jedynym wierszu standardowego wejścia znajduje się jedna liczba całkowita dodatnia, mniejsza niż 10^{30}, zapisana za pomocą co najwyżej 30 cyfr dziesiętnych. Jest to kwota jaką klient zamierza pożyczyć.

Wyjście

W każdym z dwóch wierszy standardowego wyjścia należy zapisać rosnący ciąg liczb całkowitych dodatnich z zakresu $[0 \ldots 99]$ oddzielonych odstępami albo jedno słowo `NO`:

1. pierwszy wiersz zawiera numery bankomatów, w porządku rosnącym, z których powinien skorzystać klient, by pobrać pożyczkę, albo słowo `NO`, gdy nie może jej pobrać zgodnie z ustalonymi regułami;
2. w drugim wierszu powinny znajdować się numery bankomatów, w porządku rosnącym, z których powinien skorzystać klient oddając pożyczkę, albo słowo `NO`, jeżeli jest to niemożliwe.

Przykład

Dla danych wejściowych:

```
7
```

poprawną odpowiedzią jest:

```
0 1 3 4
0 3
```

2. Skok w bok

Zadanie

Plansza do gry "Skok w bok" jest nieskończoną taśmą pól, nieograniczoną zarówno w lewo jak i w prawo. Na polach planszy stoją pionki. Ich liczba jest skończona. Na jednym polu może stać równocześnie wiele pionków. Zakładamy, że pierwsze od lewej pole, na którym jest przynajmniej jeden pionek, ma numer 0. Pola na prawo od niego są oznaczone kolejno liczbami naturalnymi 1, 2, 3 itd., a pola w lewo liczbami ujemnymi: -1, -2, -3 itd. Ustawienie pionków na taśmie, które będziemy także nazywać konfiguracją, można opisać w ten sposób, że dla każdego pola, na którym jest co najmniej jeden pionek, podaje się numer pola i liczbę pionków na tym polu. Są dwa rodzaje ruchów, zmieniających konfiguracje: skok w prawo i skok w lewo. Skok w prawo polega na zabraniu po jednym pionku z wybranych dwóch sąsiednich pól o numerach p oraz $p+1$ i dodaniu jednego pionka na polu $p+2$. Skok w lewo: zabieramy jeden pionek z pola $p+2$, a dodajemy po jednym na polach p i $p+1$. Mówimy, że konfiguracja jest końcowa, jeśli na dowolnych dwóch sąsiednich polach znajduje się co najwyżej jeden pionek. Dla każdej konfiguracji istnieje dokładnie jedna konfiguracja końcowa, którą można z niej otrzymać w wyniku skończonej liczby skoków w prawo lub w lewo.

Napisz program, który:

1. Wczytuje opis konfiguracji początkowej
2. Znajduje konfigurację końcową, do jakiej można doprowadzić daną konfigurację początkową

Wejście

W pierwszym wierszu standarowego wejścia jest zapisana jedna liczba całkowita dodatnia $1 \leqslant n \leqslant 10^4$. Jest to liczba niepustych pól danej konfiguracji początkowej. W każdym z kolejnych n wierszy znajduje się opis jednego niepustego pola konfiguracji początkowej w postaci pary liczb całkowitych p, m. Liczba $0 \leqslant p \leqslant 10^4$ to numer pola, a $1 \leqslant m \leqslant 10^8$ to liczba pionków na tym polu. Opisy są uporządkowane rosnąco względem numerów pól.

Wyjście

W pierwszym wierszu standardowego wyjścia należy zapisać numery niepustych pól konfiguracji końcowej, do której można przekształcić daną konfigurację początkową. Numery te powinne być uporządkowane rosnąco.

Przykład

Dla danych

```
2
0 5
3 3
```

(5 pionków na polu 0 i 3 pionki na polu 3) poprawnym rozwiązaniem jest

```
-4 -1 1 3 5
```

3. Bramki XOR

Zadanie

Każda bramka XOR ma dwa wejścia i jedno wyjście, a jej działanie opisuje następująca tabelka

wejście 1	wejście 2	wyjście
0	0	0
0	1	1
1	0	1
1	1	0

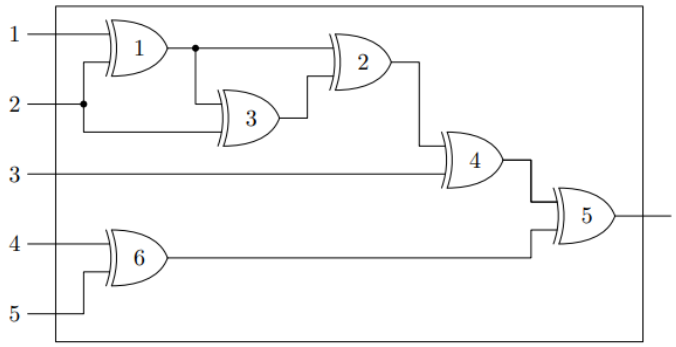
Siecią XOR nazywamy układ bramek XOR, mający n wejść i jedno wyjście, spełniający następujące warunki:

1. Każde wejście sieci XOR jest połączone z przynajmniej jednym wejściem bramki.
2. Każde wejście każdej bramki jest połączone z jednym wejściem sieci albo z jednym wyjściem innej bramki.
3. Wyjście dokładnie jednej bramki jest połączone z wyjściem sieci.
4. Każde wyjście bramki w sieci jest połączone z przynajmniej jednym wejściem innej bramki - albo z jedynym wyjściem sieci.
5. Istnieje taka numeracja bramek, że do każdego wejścia dowolnej bramki jest podłączone wejście sieci albo wyjście bramki o mniejszym numerze.

Przedstawiony na rysunku układ 6 bramek mający 5 wejść i 1 wyjście spełnia warunki 1 - 5, więc jest siecią XOR. Uwaga: bramki na rysunku zostały ponumerowane dowolnie, ale istnieje numeracja spełniająca warunek określony w punkcie 5. Wszystkie wejścia sieci są ponumerowane od 1 do n. Stan wejść sieci XOR opisuje słowo wejściowe utworzone z n cyfr dwójkowych 0 i 1 - przyjmujemy, że i-ta od lewej cyfra danego słowa wejściowego, to stan i-tego wejścia sieci. Sieci XOR będziemy testowali podając na wejściu kolejne słowa z ustalonego zakresu i zliczając liczbę otrzymanych w wyniku jedynek.

Napisz program, który:

1. wczytuje opis sieci XOR i ograniczenie zakresu, w jakim będziemy testowali sieć,
2. oblicza liczbę jedynek otrzymanych na wyjściu sieci dla słów wejściowych z danego zakresu,



Rysunek 1: Przykładowa sieć XOR

Wejście

W pierwszym wierszu standardowego wejścia są zapisane trzy liczby całkowite dodatnie. Jest to liczba wejść $3 \leqslant n \leqslant 100$ danej sieci XOR, liczba bramek $3 \leqslant m \leqslant 3000$ oraz numer bramki połączonej z wyjściem sieci. W kolejnych m wierszach znajdują się opisy połączeń bramek sieci. W i-tym z tych wierszy znajduje się opis połączeń dwóch wejść bramki o numerze i, który ma postać dwóch liczb całkowitych. Jeśli odpowiednie wejście do bramki jest połączone z wejściem do sieci o numerze k, to opisem tego połączenia jest liczba ujemna $-k$, a jeśli wejście do bramki jest połączone z wyjściem innej bramki o numerze j, to opisem tego połączenia jest liczba dodatnia j. W kolejnych 2 wierszach są zapisane dwa n-bitowe słowa (dolne i górne ograniczenie zakresu testowania sieci).

Wyjście

Na standardowe wyjście należy wypisać jedną liczbę całkowitą nieujemną - liczbę jedynek, jakie powinniśmy otrzymać na wyjściu zadanej sieci XOR dla słów wejściowych s z danego zakresu $a \leqslant s \leqslant b$, gdzie nierówność $\leqslant$ należy rozumieć jako relację porządku zgodnego z wartościami liczbowymi słów dwójkowych.

Przykład

Dla danych zawierających opis przedstawionej powyżej sieci XOR

```
5 6 5
-1 -2
1 3
1 -2
2 -3
4 6
-4 -5
00111
01110
```

poprawnym rozwiązaniem jest

```
5
```