동전 게임

영희와 동수는 동전 던지기 게임을 하고 있다. 이 게임은 K번 라운드로 구성되고 다음과 같은 규칙들을 따른다:

1. 한 라운드에서 영희와 동수는 한 번씩 동전을 던지고 항상 영희가 먼저 던진다.

2. 동전을 던져 앞면이 나오면 1점을 얻고, 뒷면이 나오면 점수를 얻지 못한다.

3. 한 명이 남은 기회에 모든 점수를 얻더라도 상대방이 현재까지 얻은 점수보다 작게 되면 게임 도중 어떤 시점에서도 게임은 바로 끝난다.

0이상 K이하인 임의의 정수 M과 N에 대해서, 이것이 항상 게임이 끝난 후 영희와 동수가 얻는 점수가 되는 것은 아니다. 예를 들어서, $K=2$인 경우에, M과 N의 모든 경우에 대해서, 이것이 영희와 동수가 얻는 점수가 될 수 있는 지의 여부는 다음 표와 같다:

M	N	영희, 동수의 점수가 될 가능성
0	0	가능
0	1	가능
0	2	불가능
1	0	가능
1	1	가능
1	2	가능
2	0	가능
2	1	가능
2	2	가능

위 표에서 영희와 동수의 점수가 0과 2가 되는 것이 불가능한 이유는 두 번째 라운드에서 영희가 뒷면이 나와서 점수를 얻지 못하는 순간 게임의 규칙 3에 의해서 0과 1로 게임이 끝나기 때문이다.

0이상 K이하인 정수 M과 N이 주어질 때, 이 두 정수가 각각 영희와 동수의 점수가 될 수 있는지 여부를 판별하는 프로그램을 작성하시오.

소스파일의 이름은 coin.c 또는 coin.cpp이며 수행시간은 1초를 넘을 수 없다. 사용하는 메모리는 128MB를 넘을 수 없다.

입력 형식

다음 정보가 표준 입력으로 주어진다. 첫 줄에 게임의 라운드 수를 나타내는 정수 $K(1 \le K \le 1{,}000)$가 주어진다. 두 번째 줄에는 입력의 개수를 나타내는 정수 C $(1 \le C \le 100{,}000)$가 주어진다. 다음 이어지는 C개의 줄 각각에는 하나의 입력을 나타내는 두 정수 M과 N $(0 \le M, N \le K)$이 주어진다.

출력 형식

다음 정보를 표준 출력으로 출력한다. 출력은 C개의 줄로 구성된다. 게임에서 영희와 동수의 점수가 각각 M과 N이 될 수 있다면 1, 아니면 0을 각 줄에 출력한다.

부분문제의 제약 조건

- 부분문제 1: 전체 점수 100점 중 9점에 해당하며, $K=2$이다.
- 부분문제 2: 전체 점수 100점 중 19점에 해당하며, $K=3$이다.
- 부분문제 3: 전체 점수 100점 중 21점에 해당하며, $K=5$이다.
- 부분문제 4: 전체 점수 100점 중 51점에 해당하며, 원래의 제약조건 이외에 아무 제약조건이 없다.

입력과 출력의 예

입력

```
5
4
5 5
5 1
0 3
1 4
```

출력

```
1
0
1
0
```

문제 1번 풀이프로그램예시

```
#include <stdio.h>
#include <algorithm>
using namespace std;

const int K_MAX = 1000;

int main()
{
	int c, k;
	scanf("%d%d", &k, &c);

	for (int i=0; i<c; i++) {
		int m, n;
		scanf("%d%d", &m, &n);

		if (m <= n && n > m + (k-m+1)/2 ||
			m >  n && m > n + (k-n)/2 + 1)
			printf("0\n");

		else
			printf("1\n");
	}

	return 0;
}
```

카드게임

지훈이는 최근에 혼자 하는 카드게임을 즐겨하고 있다. 게임에 사용하는 각 카드에는 양의 정수 하나가 적혀 있고 같은 숫자가 적힌 카드는 여러 장 있을 수 있다. 게임방법은 우선 짝수개의 카드를 무작위로 섞은 뒤 같은 개수의 두 더미로 나누어 하나는 왼쪽에 다른 하나는 오른쪽에 둔다. 그리고 빈 통을 하나 준비한다.

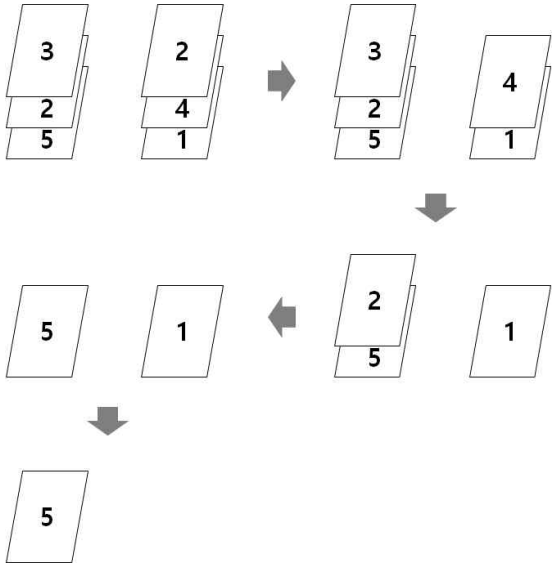
이제 각 더미의 제일 위에 있는 카드끼리 서로 비교하며 게임을 한다. 게임 규칙은 다음과 같다. 지금부터 왼쪽 더미의 제일 위 카드를 왼쪽 카드로, 오른쪽 더미의 제일 위 카드를 오른쪽 카드로 부르겠다.

(1) 언제든지 왼쪽 카드만 통에 버릴 수도 있고 왼쪽 카드와 오른쪽 카드를 둘 다 통에 버릴 수도 있다. 이때 얻는 점수는 없다.
(2) 오른쪽 카드에 적힌 수가 왼쪽 카드에 적힌 수보다 작은 경우에는 오른쪽 카드만 통에 버릴 수도 있다. 오른쪽 카드만 버리는 경우에는 오른쪽 카드에 적힌 수만큼 점수를 얻는다.
(3) (1)과 (2)의 규칙에 따라 게임을 진행하다가 어느 쪽 더미든 남은 카드가 없다면 게임이 끝나며 그때까지 얻은 점수의 합이 최종 점수가 된다.

다음 예는 세 장 씩 두 더미의 카드를 가지고 게임을 시작하는 경우이다.

카드 순서	왼쪽 더미	오른쪽 더미
1	3	2
2	2	4
3	5	1

이 경우, 우선 오른쪽 카드 2가 왼쪽 카드 3보다 작으므로 규칙 (1)에 따라 왼쪽 카드만 버리거나 왼쪽 카드와 오른쪽 카드를 모두 버리거나, 규칙 (2)에 따라 오른쪽 카드만 버릴 수 있다. 만약 오른쪽 카드만 버리는 것으로 선택하면, 2만큼 점수를 얻고 오른쪽 카드 2는 버린다. 이제 오른쪽 더미의 제일 위 카드는 4이고 이는 왼쪽 카드 3보다 크므로 규칙 (1)에 따라 왼쪽 카드만 버리거나 왼쪽 카드와 오른쪽 카드를 둘 다 버릴



수 있다. 만약 둘 다 버리는 것으로 선택하면, 이제 왼쪽 카드는 2가 되고 오른쪽 카드는 1이 된다. 이 경우 다시 규칙 (1)과 (2)에 따라 세 가지 중 한가지를 선택할 수 있고, 그 중 왼쪽 카드만 버리는 것으로 선택하면 이제 왼쪽 카드는 5가 되고 오른쪽 카드는 1이 된다. 이 경우에도 역시 규칙 (1)과 (2)에 따라 세 가지 중 한가지를 선택할 수 있고, 그 중 오른쪽 카드만 버리는 것으로 선택하면 1만큼 점수를 얻고 오른쪽 카드 1은 버린다. 이제 오른쪽 더미에는 남은 카드가 없으므로 규칙 (3)에 따라 게임이 끝나며 최종 점수는 2+1=3이 된다.

두 더미의 카드가 주어졌을 때, 게임을 통해 얻을 수 있는 **최종 점수의 최대값**을 출력하는 프로그램을 작성하시오. 위 예에서 최종 점수의 최대값은 7이다.

소스파일의 이름은 card.c 또는 card.cpp이며 수행시간은 1초를 넘을 수 없다. 사용하는 메모리는 128MB를 넘을 수 없다.

입력 형식

표준 입력의 첫 줄에는 한 더미의 카드의 개수를 나타내는 자연수 N $(1 \le N \le 2{,}000)$이 주어진다. 다음 줄에는 왼쪽 더미의 카드에 적힌 정수 A $(1 \le A \le 2{,}000)$가 카드 순서대로 N개

주어진다. 그 다음 줄에는 오른쪽 더미의 카드에 적힌 정수 $B(1 \leq B \leq 2{,}000)$가 카드 순서대로 N개 주어진다. 각 더미에는 같은 숫자를 가진 카드가 두 개 이상 있을 수 있다.

출력 형식

표준 출력에 얻을 수 있는 최종 점수의 최대값을 출력한다.

부분문제의 제약 조건

- **부분문제 1:** 전체 점수 100점 중 31점에 해당하며, $1 \leq N \leq 10$이다.
- **부분문제 2:** 전체 점수 100점 중 33점에 해당하며, $1 \leq N \leq 25$이다.
- **부분문제 3:** 전체 점수 100점 중 36점에 해당하며, 원래의 제약조건 이외의 아무 제약 조건이 없다.

입력과 출력의 예

입력(1)

```
3
3 2 5
2 4 1
```

출력(1)

```
7
```

입력(2)

```
4
1 2 3 4
4 1 2 3
```

출력(2)

```
6
```

문제 2번 풀이프로그램예시

```
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;

int N;
vector <int> player;
vector <int> enemy;
vector < vector <int> > d;

int solve(int i, int j) {
	if (i >= N || j >= N)
		return 0;

	if (d[i][j] != -1)
		return d[i][j];

	int ret = 0;
	ret = max(ret, solve(i + 1, j));
	ret = max(ret, solve(i + 1, j + 1));
	if (player[i] > enemy[j])
		ret = max(ret, enemy[j] + solve(i, j + 1));

	d[i][j] = ret;
	return ret;
}

int main(void) {
	scanf("%d", &N);

	for (int i = 0; i < N; i++) {
		int ball;
		scanf("%d", &ball);
		player.push_back(ball);
	}
	for (int i = 0; i < N; i++) {
		int ball;
		scanf("%d", &ball);
		enemy.push_back(ball);
	}

	d = vector < vector <int> >(N, vector <int> (N, -1));

	printf("%d\n", solve(0, 0));

	return 0;
}
```

트리

N개의 노드로 구성된 루트가 있는 트리가 다음과 같이 주어진다. 각 노드는 0부터 $N-1$까지의 번호로 구별되고, 0번 노드는 루트 노드이고, 나머지 노드 각각은 0번 노드의 자식 노드이다.

트리에 적용할 수 있는 연산은 세 종류이며, 이를 통해 트리의 모양을 바꾸거나 트리 에지에 색칠을 할 수 있다. 각 연산과 그 의미는 다음과 같다.

1. paint(a, b, c): a번 노드와 b번 노드를 잇는 최단 경로를 찾은 후, 경로 상에 있는 모든 에지를 색깔 c로 칠한다.
2. move(a, b): a번 노드의 부모 노드를 b번 노드로 바꾼다. 단, b번 노드는 a번 노드를 루트로 하는 부트리(subtree)에 속하지 않는다. 부모 노드를 바꾸기 전 a번 노드의 부모 노드를 p라 할 때, 새로운 에지 (a,b)는 원래의 에지 (a,p)의 색깔을 갖는다.
3. count(a, b): a번 노드와 b번 노드를 잇는 최단경로를 찾은 후, 그 경로 사이에 있는 에지에 칠해진 서로 다른 색깔의 개수를 출력한다.

에지에 칠하는 색깔 c를 정수로 표시한다. 그리고 처음에는 모든 에지의 색깔이 0이라고 가정한다.

예를 들어, 그림 1에서 보인 것처럼 6개의 노드로 구성된 초기 트리에 적용된 연산이 차례로
move(1,3); move(5,3); paint(5,4,8); move(3,4); paint(0,3,7); count(2,5);
일 때, 각 연산을 실행한 후 어떻게 트리의 모양과 에지 색깔이 바뀌는지를 아래 그림 2부터 그림 4에서 차례대로 보였다.

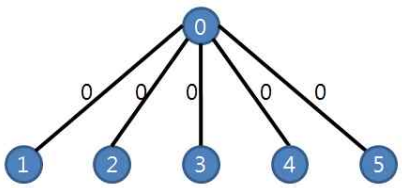


그림 2. 초기 형태

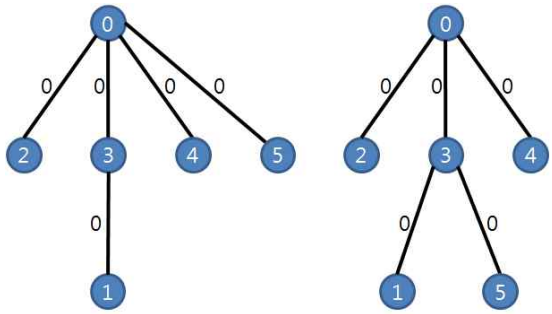


그림 3. 좌측: move(1,3)을 실행한 후
우측: move(5,3)을 실행 한 후

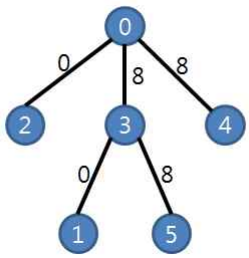


그림 4. paint(5,4,8)을 실행한 후

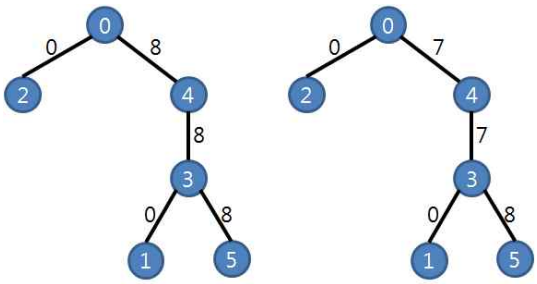


그림 5. 좌측: move(3,4)를 실행한 후
우측: paint(0,3,7)을 실행한 후

그리고, 마지막 연산 count(2,5)에 대한 결과로는 3을 출력하게 된다. 왜냐하면, 그림 4의 우측 그림에서 보듯이 2번 노드와 5번 노드 사이의 최단 경로 상에 있는 에지들에 칠해진 색깔이 {0,7,8}로 3가지이기 때문

이다.

트리에 대한 정보와 일련의 연산이 주어질 때, 각 연산을 효과적으로 실행하는 프로그램을 작성하시오.

소스파일의 이름은 tree.c 또는 tree.cpp이며 수행시간은 3초를 넘을 수 없다. 사용하는 메모리는 128MB를 넘을 수 없다.

입력 형식

다음 정보가 표준 입력으로 주어진다. 첫째 줄에는 앞에서 설명한 트리의 노드 개수를 나타내는 정수 $N(5 \le N \le 10^5)$과 연산의 개수를 나타내는 정수 $K(1 \le K \le 3 \times 10^5)$가 주어진다. 이어서 K 줄에 걸쳐 각 연산에 관한 정보가 한 줄에 하나씩 주어지는데, 각 줄에는 연산의 종류를 나타내는 정수 $r(1 \le r \le 3)$이 첫 번째로 주어진다.

$r = 1$일 경우엔 연산이 paint 임을 의미하며, 세 정수 (a,b,c)가 추가로 같은 줄에 주어지는데, 여기서 $a,b(0 \le a,b \le N-1)$는 노드 번호를, $c(0 \le c \le 10^9)$는 색의 번호를 나타낸다.

$r = 2$일 경우엔 연산이 move임을 의미하며, 두 정수 $a,b(1 \le a \le N-1,\ 0 \le b \le N-1)$가 추가로 같은 줄에 주어지는데, 이는 노드 번호를 나타낸다.

$r = 3$일 경우엔 연산이 count임을 의미하며, 두 정수 $a,b(0 \le a,b \le N-1)$가 추가로 같은 줄에 주어지는데, 이는 노드 번호를 나타낸다.

노드의 개수가 N인 트리의 초기 모양은 그림 1에서 보인 것처럼 0번 노드가 루트이고, 나머지 노드들의 부모 노드는 0번 노드이며, 초기 트리의 모든 에지 색깔은 0이라고 가정한 사실을 기억하기 바란다.

또한, paint와 count 연산 시 a번 노드와 b번 노드 사이의 최단경로의 길이는 항상 $1{,}000$ 이하이다.

출력 형식

다음 정보를 표준 출력으로 출력한다. 입력에서 주어진 count 연산 각각에 대해, 그 순서대로 그 때의 결과 값을 한 줄에 출력한다.

부분문제의 제약 조건

- **부분문제 1**: 전체 점수 100점 중 8점에 해당하며, move 연산(즉, $r = 2$인 경우)은 없으며, $5 \le N \le 10$이고 $1 \le K \le 5$이다.
- **부분문제 2**: 전체 점수 100점 중 18점에 해당하며, $5 \le N \le 1{,}000$이고 $1 \le K \le 2{,}000$이며, paint와 count 연산에서 주어지는 인수 a, b에서, b번 노드는 항상 a번 노드와 루트를 잇는 최단경로(a번 노드와 루트를 포함하는 경로) 상에 있다.
- **부분문제 3**: 전체 점수 100점 중 28점에 해당하며, $5 \le N \le 1{,}000$이고 $1 \le K \le 2{,}000$ 이다.
- **부분문제 4**: 전체 점수 100점 중 46점에 해당하며, 원래의 제약조건 이외에 아무 제약조건이 없다.

입력과 출력의 예

입력(1)

```
6 8
2 1 3
2 5 3
1 5 4 8
3 4 5
2 3 4
1 0 3 7
3 2 5
3 4 2
```

출력(1)

```
1
3
2
```

입력(2)

```
7 15
2 3 2
2 4 3
2 5 3
2 6 2
3 1 6
1 3 3 2
1 1 6 5
1 4 2 3
1 2 5 4
1 2 0 7
3 4 6
3 4 1
2 3 2
3 2 2
3 5 6
```

출력(2)

```
1
3
4
0
2
```

문제 3번 풀이프로그램예시

```
#include <cstdio>
#include <algorithm>
using namespace std;
#define MN 300005
//n : |V|, m : |E|, u : parent
int n, m, u[MN];

void input(void) {
	scanf("%d%d",&n,&m);
	u[0] = -1;
	for (int i = 1; i <= n-1; i++) u[i] = 0;
}

//d : vertex color, c : union set, K : union number, s : counting color
int d[MN], c[MN], K, s[MN];

int fndpar(int A, int B) {
	++K;
	int pA, pB, r;
	pA = A, pB = B;
	while (pA >= 0 || pB >= 0) {
		if (pA >= 0) {
			if (c[pA] == K) { r = pA; break; }
			c[pA] = K, pA = u[pA];
		}
		if (pB >= 0) {
			if (c[pB] == K) { r = pB; break; }
			c[pB] = K, pB = u[pB];
		}
	}
	return r;
}

struct Query {
	int type, A, B, C;
} q[MN];

int _c[MN], cn;

int fnd(int X) {
	int L = 1, R = cn, M;
	while (L <= R) {
		M = (L+R)/2;
		if (_c[M] == X) break;
		if (_c[M] > X) R = M-1;
		else L = M+1;
	}
	return M;
}
```

```
void solve(void) {
	for (int i = 1; i <= m; i++) {
		scanf("%d",&q[i].type);
		if (q[i].type == 1) scanf("%d%d%d",&q[i].A,&q[i].B,&q[i].C), _c[++cn] = q[i].C;
		else scanf("%d%d",&q[i].A,&q[i].B);
	}
	sort(_c+1, _c+cn+1);
	int type, A, B, C, CA, p;
	for (int i = 1; i <= m; i++) {
		type = q[i].type;
		if (type == 1) { //coloring
			A = q[i].A; B = q[i].B;
			if (q[i].C == 0) C = 0;
			else C = fnd(q[i].C);
			if (A == B) continue;
			CA = fndpar(A, B);
			while (A >= 0 && CA != A) { if (A) d[A] = C; A = u[A]; }
			while (B >= 0 && CA != B) { if (B) d[B] = C; B = u[B]; }
		} else if (type == 2) { //cut - link
			A = q[i].A; B = q[i].B;
			if (A == B) continue;
			u[A] = B;
		} else { //asking
			A = q[i].A; B = q[i].B;
			if (A == B) { printf("0\n"); continue; }
			CA = fndpar(A, B);
			int res = 0;
			while (A >= 0) {
				if (CA == A) break;
				if (s[d[A]] != K) { res++; s[d[A]] = K; }
				A = u[A];
			}
			while (B >= 0) {
				if (CA == B) break;
				if (s[d[B]] != K) { res++; s[d[B]] = K; }
				B = u[B];
			}
			printf("%d\n",res);
		}
	}
}

int main(void) {
	input();
	solve();
	return 0;
}
```

미술관

K 미술관은 많은 벽으로 구성된 특이한 구조를 가진 건축물로 유명하다. 미술관의 내부 조명을 위해서 두 개의 전등이 한 쪽 벽의 양 끝 모서리에 설치되어 있는데, 건물의 내부에 조명이 미치지 않는 곳이 없다. 즉, 건물 내부의 모든 장소는 적어도 하나의 전등으로부터 조명을 받을 수 있다.

정보올림피아드를 준비하는 홍길동은 이 미술관 건물을 좋아해서 시간이 날 때마다 관람하러 온다. 하루는 미술관을 관람하던 중에 갑자기 "미술관 내부의 두 지점을 연결하는 최단 경로는 어떤 모양일까?"라는 의문점이 떠올랐다. 일반적인 다각형에서 최단 경로 알고리즘을 구현하는데 힘들었던 기억을 되살리면서, 두 개의 전등으로 모든 곳을 비출 수 있는 미술관의 특이한 구조 때문에 최단 경로를 쉽게 구할 수 있지 않을까라는 생각을 하게 되었다.

미술관을 n개의 정점을 가진 다각형 $P=(v_0,v_1,\ldots,v_{n-1})$로 나타낼 수 있다. 정점 리스트는 다각형의 경계선을 반시계방향으로 따라가면서 정점들을 순서대로 나열한 것이다. 미술관에서 전등이 설치된 장소를 정점 v_0과 v_1이라고 하자. 에지 (v_0,v_1)은 수평 선분으로 v_0의 x-좌표는 항상 v_1의 x-좌표보다 작다. v_0과 v_1을 제외한 나머지 모든 정점의 y-좌표는 v_0의 y-좌표보다 크다(그림 1 참조).

미술관 내부의 어떤 장소 q가 전등 v의 조명을 받는다는 것은, 두 점 q와 v를 연결하는 선분이 P의 외부와 만나지 않는다는 것을 말한다. P의 모든 점은 v_0 또는 v_1로부터 조명을 받는다는 사실에 유의하라.

그림 1의 다각형에서 정점 v_8과 v_{11}은 v_1로부터만 조명을 받고, v_3과 v_4는 v_0으로부터만 조명을 받는다. 나머지 정점들은 v_0과 v_1 둘 다로부터 조명을 받는다. 두 정점 사이의 최단 경로가 항상 다각형의 정점에서만 꺾인다는 것은 잘 알려져 있다. 예를 들어, 두

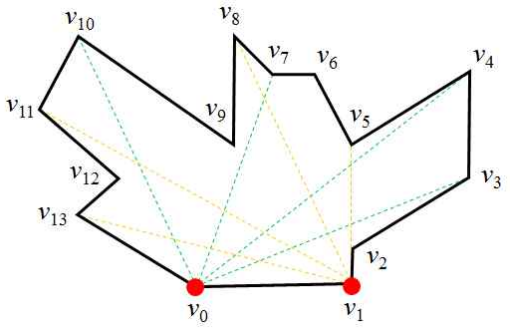


그림 6. 다각형의 모든 점이 v_0 또는 v_1로부터 조명을 받는다.

정점 v_4와 v_{11} 사이의 최단 경로는 (v_4,v_5,v_9,v_{11})이다. 두 정점 v_5와 v_1 사이의 최단 경로는 하나의 선분인 (v_5,v_1)이다.

홍길동을 도와서 다각형 P의 두 정점이 주어질 때, 두 정점 사이의 최단 경로를 구하는 프로그램을 작성하시오.

소스파일의 이름은 gallery.c 또는 gallery.cpp이며 수행시간은 1초를 넘을 수 없다. 사용하는 메모리는 128MB를 넘을 수 없다.

입력 형식

다음 정보가 표준 입력으로 주어진다. 첫째 줄에 다각형 P의 정점의 개수를 나타내는 정수 n이 주어진다. n은 3 이상 100,000 이하이다. 둘째 줄부터 n개의 줄에는 v_0으로부터 시작하여 각 줄마다 하나씩 P의 각 정점 v_i의 좌표를 나타내는 두 개의 정수가 주어진다($i=0,1,\ldots,n-1$). 각 좌표는 -10^9 이상 10^9 이하이다. P의 모든 점은 v_0 또는 v_1로부터 조명을 받고, v_0과 v_1의 y-좌표는 같으며, v_0은 v_1보다 x-좌표가 작다. v_0과 v_1을 제외한 나머지 모든 정점은 v_0보다 y-좌표가 크다. P의 경계선을 따

라서 연속된 어떤 세 정점도 일직선 상에 위치하지 않는다. 마지막 줄에는 최단 경로를 구하려고 하는 두 정점 v_i와 v_j를 나타내는 정점 번호인 정수 i와 j가 주어진다($i \neq j$). 여기서 v_i는 출발점이고 v_j는 도착점이다.

출력 형식

다음 정보를 표준 출력으로 출력한다. 입력으로 주어진 두 정점 v_i와 v_j를 연결하는 최단 경로를 $(w_0, w_1, \ldots, w_{m-1})$이라고 하자. 여기서 $w_0 = v_i$, $w_{m-1} = v_j$이고, $w_k (1 \leq k \leq m-2)$는 최단 경로 상의 꺾인 점이다. 첫째 줄에 m을 출력하고, 둘째 줄에 w_k에 해당하는 P의 정점 번호를 순서대로 출력한다($0 \leq k \leq m-1$).

부분문제의 제약 조건

- **부분문제 1**: 전체 점수 100점 중 22점에 해당하며, P의 모든 점이 정점 v_0으로부터 조명을 받을 수 있고, $n \leq 500$ 이다.
- **부분문제 2**: 전체 점수 100점 중 29점에 해당하며, P의 모든 점이 정점 v_0으로부터 조명을 받을 수 있고, $n \leq 100{,}000$ 이다.
- **부분문제 3**: 전체 점수 100점 중 13점에 해당하며, $n \leq 100$ 이다.
- **부분문제 4**: 전체 점수 100점 중 16점에 해당하며, $n \leq 500$ 이다.
- **부분문제 5**: 전체 점수 100점 중 20점에 해당하며, 원래의 제약조건 이외의 아무 제약 조건이 없다.

입력과 출력의 예

입력(1) [그림 1 참조]

```
14
5 2
9 2
9 3
12 5
12 8
9 6
8 8
7 8
6 9
6 6
2 9
1 7
3 5
2 4
4 11
```

출력(1)

```
4
4 5 9 11
```

입력(2) [모든 점이 v_0으로부터 조명을 받는 예]

```
9
4 2
10 2
12 5
8 4
9 7
6 7
4 5
3 6
2 6
5 2
```

출력(2)

```
3
5 3 2
```

문제 4번 풀이프로그램예시

```
#include <stdio.h>
#include <vector>
#include <algorithm>
using namespace std;

struct point
{
	long long x, y;
	point(){}
	point(long long x, long long y) : x(x), y(y) {}
};


const int N_MAX = 100000;
int n;
point p[2 * N_MAX];
int q0, q1;

int ccw(const point& a, const point& b, const point& c)
{
	long long r = (b.x-a.x)*(c.y-a.y) - (c.x-a.x)*(b.y-a.y);
	return r>0 ? 1 : r<0 ? -1 : 0;
}

vector<int> getPath(int s, int e)
{
	if (s == e)
		return vector<int>(1, s);

	vector<int> res;

	if (s+1 == e) {
		res.push_back(s);
		res.push_back(e);
		return res;
	}

	if (s == 0) {
		res = getPath(e, n);
		res.back() -= n;
		return res;
	}

	const point& dest = p[e];

	res.push_back(s);

	for (int i=s; i<e; i++) {
		while (res.size() >= 2) {
			const int s = res.size();
			const point& p0 = p[ res[s-2] ];
			const point  p1 = p[ res.back() ];

			if (ccw(p0, p1, p[i+1]) >= 0)
				res.pop_back();

			else
				break;
		}
		res.push_back(i+1);
	}

	return res;
}

int main()
{
	scanf("%d", &n);
	for (int i=0; i<n; i++) {
		long long x, y;
		scanf("%lld%lld", &x, &y);

		p[n+i] = p[i] = point(x, y);
	}
	scanf("%d%d", &q0, &q1);

	vector<int> ans = getPath(min(q0, q1), max(q0, q1));
	if (ans[0] != q0)
		reverse(ans.begin(), ans.end());
```

```
        printf("%d\n", ans.size());
        for (int i=0; i<ans.size(); i++)
                printf("%d ", ans[i]);
        printf("\n");

        return 0;
}
```