Problem A. Dungeon 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

당신은 Just Ordinary Inventions사를 아는가? 이 회사의 업무는, "그저 평범한 발명(just ordinary inventions)"를 하는것이다.

JOI군은, Just Ordinary Invention사가 개발한 최신 게임을 하고 있다.

이 게임은, 몇개의 방과 몇개의 길로 구성된 던전을 탐험하는 게임이다. 길은 던전 내의 서로 다른 2개의 방을 연결하고 있으며, 양방향으로 이동 가능 하다. 어떤 다른 2개의 방에 대해서도, 그들을 연결하는 길은 최대 하나 밖에 없고, 양쪽에 같은 방이 있는 길은 존재하지 않는다, 또한 던전에 있는 어떤 2개의 방도, 몇개의 길을 이용하면 서로 이동할 수 있다는 것을 알고 있다. 각 방끼리는 매우 유사하며, 동일한 갯수의 길이 나오는 방 끼리는,방 모습을 보는 것 만으로는 구별할 수 없다.

이 게임에서는 공략을 돕기 위해 각 방마다 표적과 받침대가 준비되어있다. 방에서 나오는 길은, 표적을 기준으로 첫번째, 두번째, $\cdots$로 셀 수 있다. 게임 중 던전의 구조가 바뀌지는 않는다. 따라서, 같은 방에서 같은 번호의 길을 통해 이동하면 항상 같은 방에 도착한다. 받침대는 플레이어가 색을 변경할 수 있는 구슬이 하나 있다. 보석의 색은 색 1, 색 2, $\cdots$, 색 X중 하나 이며, 게임 시작시 각 방 보석의 색은 색 1이다. 보석의 색은 플레이어가 색을 변경하지 않는 한 바뀌지 않는다.

JOI군은, 만약 던전에 구조, 즉 던전의 방들이 어떻게 길로 연결되어 있는가에 대해 알면, 이 게임은 간단히 공략이 될 것이라 느꼈다. 하지만, JOI군이 여러가지로 시도해봐도 던전의 구조를 알 수 없었다. 그래서 당신은 JOI군을 대신해서 던전의 구조를 결정하는 프로그램을 작성하기로 했다.

던전을 탐험하고, 던전의 구조를 결정하는 프로그램을 작성하여라. 단, JOI군은 던전의 구조를 완전히 아는것을 원치 않았기 때문에, 프로그램은 던전의 구조를 직접 답하는 대신, 1이상 R이하의 정수 i에 대해, "최소 i개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍이 몇개 있는가" (단, 방의 순서만 바꾼 것은 같은 쌍으로 본다.)를 답해야 한다. 던전을 탐험하기 위해, 당신에게는 다음의 라이브러리가 제공된다.

- 현재 있는 방에서, 나가는 길이 몇개 있는지를 알 수 있다.
- 현재 방 받침대에 장식되어 있는 보석의 색을 알 수 있다.
- 현재 방 받침대에 장식되어 있는 보석의 색을 지정한 색으로 바꿀 수 있다. (같은 색으로도 바꿀 수 있다.) 그 후, 방에서 나와, 길을 하나 선택하고, 그 길을 이용해 다른 방으로 이동한다.
- 마지막으로 사용한 길이, 현재 있는 방에서 몇번째 길인지 알 수 있다.

Interaction Protocol

당신은, JOI군에게 답할 방법을 담은 1개의 프로그램을 작성해야 한다. 프로그램은 `dungeon2.h`를 `include` 해야 한다. 프로그램에는, 다음의 함수가 구현되어 있어야 한다.

- `void Inspect(Int R)` 이 함수는, 처음 1번만 실행된다.
  - 인자 `R`은, 1이상 R이하의 각 정수 i에 대해, "최소 i개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍이 몇개 있는가" (단, 방의 순서만 바꾼 것은 같은 쌍으로 본다.)를 답해야 하는 것을 의미한다.

또한, 당신은 다음 함수를 불러 질문에 대한 답을 해야 한다.

- `void Answer(int D, int A)`
  - 인자 `D`, `A`는, 이 함수 호출을 할 때, "최소 D개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍은 A개가 있다."라는 답을 의미한다.

단, Answer를 호출 할 때는, 다음의 조건을 만족해야 한다.

- D는 1이상 R이하의 정수여야 한다. 이것을 만족하지 않을 경우 **오답** [1]이 된다.
- Answer를 같은 인자 D로 2번 이상 호출하면 안된다. 이것을 만족하지 않은 경우 **오답** [2]가 된다.
- Answer는 정확히 R번 호출되어야 한다. 이것을 만족하지 않은 경우, **오답** [3]이 된다.
- A는 최소 D개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍의 갯수여야 한다. 이것을 만족하지 않은 경우 **오답** [4]가 된다.

추가로, 프로그램 중에 다음 함수를 호출할 수 있다.

- `void Move(int I, int C)`
  - 인자 I는, 플레이어가 이동하기 위해서 고른 길의 번호를 의미한다. 이 함수가 호출된 직후, 플레이어는 현재 있는 방에서 I번째 길을 이용해 다른 방으로 이동한다.
  - 인자 C는, 방을 이동하기 전에, 현재 방 받침대에 있는 보석의 색을 색 C로 바꾸는 것을 의미한다.

  단, Move를 호출할 때, 다음 조건을 만족해야 한다.
  - 인자 I는, 플레이어가 현재 있는 방에서 나올 수 있는 길의 수를 K라 할 때, 1 이상 K이하의 정수여야 한다. 이것을 만족하지 않은 경우 **오답** [5]가 된다.
  - 인자 C는, 보석의 색의 종류가 X종류라고 할 때, 1 이상 X이하의 정수여야 한다. X는 Subtask에 따라 결정된다. 이것을 만족하지 않은 경우 **오답** [6]이 된다.
  - Move를 1 500 000번 이상 호출해서는 안된다. 이것을 만족하지 않은 경우 **오답** [7]이 된다.
- `int NumberOfRoads()`
  - 이 함수는, 플레이어가 현재 있는 방에서 나가는 길의 갯수를 반환한다.
- `int LastRoad()`
  - 이 함수는, 플레이어가 마지막에 사용한 길을, 현재 있는 방에서 몇번째 길인지를 반환한다. 단, Move가 한번도 호출되지 않았을 경우, -1을 반환한다.
- `int Color()`
  - 이 함수는, 플레이어가 현재 있는 방에 위치해있는 보석의 색을 반환한다.

함수 Inspect를 호출 한 후, 답에 대한 판단을 한다. 내부 사용을 위해서 다른 함수를 구현하거나, 글로벌 변수를 선언하는것은 자유이다. 하지만, 당신의 제출은 표준 입출력이나, 다른 함수에 접근하면 안 된다.

작성한 프로그램을 테스트하기 위한 채점 프로그램 샘플이, 콘테스트 사이트에서 다운로드 받을 수 있는 아카이브 안에 있다. 이 아카이브는, 제출해야하는 파일의 샘플도 들어있다.

채점 프로그램 샘플은 1개의 파일이다. 이 파일은 grader.c 혹은 grader.cpp이다. 작성한 프로그램을 테스트 하기 위해서는, 다음의 커맨드를 실행한다.

- C의 경우 `gcc -std=c11 -O2 -o grader grader.c dungeon2.c -lm`
- C++의 경우 `g++ -std=c++11 -O2 -o grader grader.cpp dungeon2.cpp`

컴파일에 성공하면, grader라는 이름의 파일이 생성된다.

실제의 채점 프로그램은, 채점 프로그램 샘플과는 다르므로 주의한다. 채점 프로그램의 샘플은 단일 프로세스로 실행된다. 이 프로그램은, 표준입력에서 입력을 받아서, 표준출력으로 결과를 출력한다.

채점 프로그램 샘플은, 표준입력에서 다음과 같은 데이터를 읽는다.

Input

- 첫째 줄에는, 정수 N, X, R이 공백으로 구분되어 들어온다. 이것은, 던전이 방 1, 방 2, $\cdots$, 방 N의 N개의 방으로 되어 있고, 보석의 색은 X종류 이며, 프로그램이 답해야 하는 값이 R개 임을 의미한다.
- 다음 $2N$개의 줄의 $2i-1$번째 줄 $(1 \leq i \leq N)$에는, 정수 D_i가 들어오고, 방 i에서 D_i개의 나가는 길이 있다는 것을 의미한다. $2i$번째 줄에는 D_i개의 정수 $T_{i1}, T_{i2}, \cdots, T_{iD_i}$가 공백으로 구분되어 들어온다. 이것은, 방 i에서 나가는 j $(1 \leq j \leq D_i)$번째 도로를 써서 이동한 방의 번호가 T_ij라는 것을 의미한다.
- 다음 R개의 줄의 j번째 줄 $(1 \leq j \leq R)$에는, 정수 A_j가 쓰여 있다. 이것은, "최소 j개의 길을 이용해서 2개의 방을 이동할 수 있는 쌍은 A_j개가 있다."는 것을 의미한다. 즉, 각 $j(1 \leq j \leq R)$에 대해, `Answer`의 인수 `D`를 j, 인수 `A`를 A_j로 해서 호출 한 경우, 채점 프로그램이 정답으로 생각하고, 다른 경우 오답으로 생각한다는 것을 의미한다.

채점 프로그램은, 플레이어의 초기 위치를 방 1로 하여, 당신이 작성한 함수를 호출한다.

Output

프로그램의 실행이 정상적으로 종료된 경우, 채점 프로그램 샘플은 표준출력으로 다음의 정보를 첫째 줄에 출력한다. (따옴표는 출력되지 않는다.)

- 정답일 경우, 함수 `Move`를 호출한 횟수가 `"Accepted : #move = 8"`처럼 출력한다.
- 오답일 경우, 오답의 종류를 `"Wrong Answer [1]"`처럼 출력한다.

Constraints

모든 입력데이터는 다음의 조건을 만족한다. N, D_i, T_{ij}의 의미에 대해서는 채점 프로그램의 샘플 입력을 참고하여라.

- $2 \leq N \leq 200$
- $3 \leq X \leq 100$
- $1 \leq R \leq 200$
- $1 \leq D_i \leq N-1$ $(1 \leq i \leq N)$
- $1 \leq T_{ij} \leq N$ 이며 $T_{ij} \neq i$ $(1 \leq i \leq N,\ 1 \leq j \leq D_i)$
- $T_{i1}, T_{i2}, \cdots, T_{iD_i}$는 서로 다르다.
- 각 i, j $(1 \leq i \leq N,\ 1 \leq j \leq D_i)$에 대해, $T_{T_{ij}k} = i$를 만족하는 k $(1 \leq k \leq D_{T_{ij}})$가 존재한다.
- 어떤 2개에 방에 대해서도, 몇개의 길을 쓰면 서로 이동할 수 있다.

이하에서, 입력데이터의 방의 수를 N, 길의 수를 M이라 한다.

Subtask 1 (17 points)

다음의 조건을 만족한다.

- $N \leq 50$
- $M \leq 100$
- $X = 100$

Subtask 2 (50 points)

다음의 조건을 만족한다.

- $N \leq 50$
- $M \leq 100$
- $X = 3$

Subtask 3 (40 points)

다음의 조건을 만족한다.

- $X = 3$을 만족한다.

이 Subtask에서는 다음에 따라 점수가 정해진다.

- 이 Subtask의 모든 테스트데이터에 대해, 다음의 최댓값을 L이라 한다.
  - `Move`의 호출 횟수를 C라고 할 때, $\frac{C}{M}$
- 이 때, 이 Subtask의 득점은
  - $L \leq 14$일 때, 56점.
  - $14 < L \leq 32$일 때, $\lfloor 70 - L \rfloor$점.
  - $32 < L \leq 64$일 때, $\lfloor 54 - \frac{L}{2} \rfloor$점.
  - $64 < L$일 때, 0점.

  여기서 $\lfloor x \rfloor$는 x를 넘지 않는 최대의 정수로 한다.

채점 시스템 상에서, 프로그램이 정상적으로 종료되었을 경우 채점 결과를 `Accepted`로 표시한다. 단 Subtask 3에서 $64 < \frac{C}{M}$인 테스트케이스에 대해서는, 결과가 **오답**으로 표시됨에 주의하라.

Example

다음은 채점 프로그램 샘플이 입력받은 입력 예제와, 그에 대응되는 함수 호출 예이다.

standard input	interaction	
	Function call	Return value
`4 3 3`	`Inspect(3)`	
`1`	`NumberOfRoads()`	`1`
`2`	`LastRoad()`	`-1`
`3`	`Move(1, 2)`	
`1 3 4`	`Color()`	`1`
`2`	`LastRoad()`	`1`
`2 4`	`NumberOfRoads()`	`3`
`2`	`Move(1, 3)`	
`2 3`	`Color()`	`2`
`4`	`Answer(1, 4)`	
`2`	`Answer(2, 2)`	
`0`	`Answer(3, 0)`	

이 예제의 함수 호출은, 반드시 의미 있는 호출은 아니라는 점에 주의하라.