1. 에덴동산(10점)

문제

곰두리가 사는 세상에서 도달할 수 없는 상상 속의 세계를 '에덴동산'이라고 부른다. 이 에덴동산이 다른 의미로 사용될 때도 있는데, 바로 어떤 방식으로도 도달할 수 없는 지역을 가리킬 때다. 이 에덴동산은 길이 이어져있지 않아서 어떤 방식을 사용해도 방문할 수가 없다.

그렇기에 만약 방문하려는 곳이 에덴동산인지 알지 못한다면 그 지역까지 도달하는 방법만 찾다가 결국 지쳐 쓰러질 수도 있다. 자신의 친구들이 에덴동산으로 추정되는 곳을 계속 찾아다니다가 지치는 것을 본 곰두리는 지명을 입력받으면 해당 지역이 에덴동산인지 판별해주는 프로그램을 만들기로 결심한다.

입력

첫 번째 줄에 입력할 길의 개수 n과 검사할 지역 수 k를 입력한다. 그리고 두 번째 줄부터 2개의 지역을 입력한다. 이렇게 입력된 지역은 서로 연결된 지역이고 연쇄적으로 적용된다. 길의 입력이 모두 끝나면 k만큼 기준점과 에덴동산인지 판별할 목표지역을 입력한다. 검사할 때, 저장되지 않은 지명을 입력하면 무조건 '에덴동산'으로 출력한다.

예를 들어보자, A B, B C를 입력하면 A와 B, 그리고 B와 C는 서로 연결되어있다. 그렇다면 A와 C 또한 연결되어 있다고 볼 수 있다. 그런데 만약 A B, C D로 입력한다면 A와 B에서 C와 D로 이동하는 것은 불가능하다. 이 때 A의 입장에서 C와 D는 에덴동산이다.

입력예시는 다음과 같다.

```
4 2
A B
B C
A D
F G
A G
C D
```

출력

기준점에서 목표지역이 에덴동산인지 판별한다. 만약 에덴동산이면 '에덴동산'을 출력하고 도달이 가능한 지역이면 '도달가능'을 출력한다.

입력예시에 따른 출력예시는 다음과 같다.

- 에덴동산
- 도달가능

2. 발표자는 누구? (15점)

문제

나쁜 곰돌 교수는 매일매일 발표할 학생을 뽑아 그 다음 날 수업에 발표를 시키는 것을 즐긴다. 이 곰돌 교수는 다음 수업에서 발표를 하게 될 사람을 특이한 방식으로 뽑는다.

이 교수가 수업에서 발표자를 뽑는 방식은 다음과 같다.

- n명의 학생이 있다고 가정하자. 이 학생들은 1부터 n까지의 숫자를 부여받고 동그랗게 모인다.
- 교수가 임의의 숫자 k를 정한다.
- 1번부터 숫자를 세기 시작한다. 그리고 k번째 학생은 제외된다.
- 이렇게 되면 n-1명의 학생이 남게 된다. 제외된 학생의 다음 학생부터 숫자를 세기 시작해서 k번째 학생을 또 제외한다.
- 이 과정을 반복했을 때 마지막으로 남은 학생은 발표자가 된다.

예를 들어 7명의 학생이 존재하고, 교수가 지정한 숫자 k가 3이라면 다음 과정으로 발표자가 정해지게 된다.

1. 1, 2, 3. 1번 학생부터 3을 센다. 3번 학생이 제외된다. 남은 학생은 6명이다.
2. 4, 5, 6. 4번 학생부터 3을 센다. 6번 학생이 제외된다. 남은 학생은 5명이다.
3. 7, 1, 2. 7번 학생부터 3을 센다. 2번 학생이 제외된다. 남은 학생은 4명이다.
4. 4, 5, 7. 3번 학생과 6번 학생은 이미 제외되었다. 4번 학생부터 3을 센다. 즉 7번 학생이 제외된다. 남은 학생은 3명이다.
5. 위의 조건으로 학생들을 세는 것을 반복한다. 1, 4, 5. 5번 학생이 제외된다. 남은 학생은 2명이다.
6. 위의 조건으로 학생들을 세는 것을 반복한다. 1, 4, 1. 1번 학생이 제외된다. 남은 학생은 1명이다.
7. 마지막으로 남은 4번 학생이 발표자가 된다.

과연 다음 수업에서 눈물의 발표를 하게 될 학생은 누구일까? 억울하게 발표하는 학생이 생기지 않게 잘 코딩해보자.

입력

우선 테스트케이스의 개수를 입력 받는다.
다음 줄부터 테스트케이스 수 만큼 학생 수 n과 교수가 지정할 수 k를 입력받는다. n과 k는 스페이스로 구분한다.

그리고 다음 줄부터 n명의 학생 이름을 입력받는다. 학생 이름들 또한 스페이스로 구분한다. 여기서 입력된 학생들은 순서대로 1부터 n까지 번호를 부여받는다.
예를 들어, A B C D E F를 학생 이름으로 입력한다면 A가 1, B가 2, C가 3, …, F가 6의 번호를 부여받는다.

입력의 조건은 다음과 같다.

- 학생 수 n는 모든 테스트케이스에 대하여 $n > 5$를 만족한다.
- 교수가 지정하는 수 k는 모든 테스트케이스에서 $k > 1$를 만족한다.
- 학생의 이름은 중복되지 않는다. (다음 장에서 계속)

- 중복되지 않은 학생의 이름 n개가 모두 입력되어야 한다.
- 오늘 발표를 한 학생은 따로 고려하지 않는다. (이미 발표한 학생을 제외하지 않고 추첨한다. 즉 오늘 발표한 학생이 내일도 발표를 하게 될 수도 있다.)

입력예시는 다음과 같다.
2
7 3
한놈 두시기 석삼 너구리 오징어 육개장 칠면조
7 3
한놈 두시기 석삼 너구리 오징어 육개장 칠면조

출력

발표자가 된 학생의 이름과 그 학생이 부여받은 번호를 출력한다. 부여받은 번호와 학생의 이름은 스페이스로 구분한다.

입력예시로 입력하면 다음 결과가 출력된다.
- 4 너구리
- 4 너구리

3. 검산 (20점)

문제

여러분들도 잘 알고 있듯, 곱셈의 우선순위는 덧셈의 우선순위보다 높다. 즉, 곱셈이 덧셈보다 뒤에 위치하더라도 곱셈부터 계산하고 나서 덧셈이 차례대로 계산된다.

예를 들어볼까? 4*8+3=35, 10*3+3*4*10+5=155은 옳게 계산되었다. 그렇다면, 3+4*8=56은 어떨까? 이것은 덧셈과 곱셈의 우선순위를 고려하지 않은 틀린 계산이다.

이렇게 식과 답을 입력받고, 해당 답이 옳게 계산된 답인지 판별해주는 검산 프로그램을 만들어보자. 이 프로그램은 = 기호와 답까지 포함된 식을 받고 해당 답이 정답인지를 판별한다.

입력

첫 번째 줄에 테스트케이스의 개수를 입력받는다. 그리고 두 번째 줄부터 테스트케이스 개수에 맞게 식을 입력받는다.

식의 조건은 다음과 같으며, 이 조건을 준수하지 않은 식은 고려하지 않아도 된다.

- 식은 스페이스가 존재하지 않으며 한 개 이상의 연산자가 존재한다.
- 피연산자와 정답은 정수로 취급한다. 소수는 입력받지 않는다.
- 연산자의 좌우에는 무조건 피연산자가 존재해야한다. (+1*3, *3+2, *+1 과 같은 식은 존재할 수 없다.)
- 모든 피연산자 x는 0 <= x <= 10를 성립한다.
- 연산자는 +, *만 존재한다.

입력예시는 다음과 같다.

2
4*8+3*10*9+4=306
10*0+5+5*0+5=5

출력

연산 우선순위를 고려한 식의 계산 결과를 출력한다. True와 False는 대소문자를 구분한다. 만약 False를 출력하면 스페이스로 한 칸을 띄우고 옳게 계산된 결과를 출력하면 된다.

입력예시에 따른 출력 예시는 다음과 같다.

- True
- False 10

(참고: 출력 결과가 2,147,483,647을 초과하는 식을 채점에 넣지 않았습니다.)

4. 곰두리의 마법대결 (25점)

문제

마법사인 곰두리는 흑마법사인 흑곰돌을 겨우 봉인시키는데 성공했다. 하지만 사악한 흑곰돌은 자신이 봉인되기 전에 곰두리에게 저주를 퍼붓기 시작했다. 스치기만해도 치명적인 흑곰돌의 저주를 곰두리는 '마법 영창'을 통해 이겨내야만 한다. 곰두리가 흑곰돌의 저주를 이겨낼 수 있게 '마법 영창'을 한번 구현해보자.

마법 영창은 다음 조건이 성립된다.

- 주문이 시간 순서대로 여러개 입력된다.
- 그리고 입력된 주문의 역순서대로 논리가 전개된다.
- 예를 들어 A B C J K N 순으로 주문이 입력 되었으면 A가 가장 처음에 발생한 주문이고, N이 가장 마지막에 발생한 주문이다. 그리고 주문의 발동은 N K J C B A 순으로 전개된다.
- 입력된 주문은 미래에 발생할 주문에 영향을 미치지 못한다. 예를 들어 A B C D로 주문이 입력 되었을 때, 주문 B는 주문 A에 영향을 끼칠 수 있지만 C D 주문에는 영향을 끼칠 수 없다.

흑곰돌이 사용하는 주문은 다음과 같다.

- 곰돌다운 - 이 주문을 막지 못하면 곰두리가 사망한다.
  - 마법 영창 중에서 '곰돌다운'이 하나 이상 남아있다면 '곰돌다운'을 막지 못한 것으로 가정해 곰두리가 사망한다.
- 무자비 - 앞에 사용된 모든 주문 3개를 지운다.
  - 예를 들어, 'A B C D 무자비' 순서대로 마법 영창을 입력하면 '무자비'의 효과로 B C D가 제거되어 A만 발동한다.
  - 무자비는 너무 무자비해서 흑곰돌이 스스로 사용한 주문마저 지워 버린다. 예를 들어, 'A B C D 곰돌다운 곰돌다운 무자비' 순서대로 마법 영창을 입력하면 'D 곰돌다운 곰돌다운'모두 제거되어서 'A B C'만 주문이 발동된다.

곰두리가 사용하는 주문은 다음과 같다.

- 어쩔티비 - 흑곰돌의 주문이 이 마법을 발동하기 직전 순서에 있다면 그 주문을 지운다.
  - 만약 'A B C D 무자비 어쩔티비' 순서대로 마법 영창을 입력하면 '어쩔티비'가 '무자비'를 지워버려 'A B C D'가 발동한다.
  - 만약 '곰돌다운 곰돌다운 어쩔티비' 순서대로 마법 영창을 입력하면 '어쩔티비'가 '곰돌다운' 하나를 지워버리지만 '곰돌다운' 하나가 남게 되어 곰두리는 사망하게 된다.
  - '어쩔티비'는 직전 마법에만 영향을 끼친다. 예를 들어 '곰돌다운 무자비 어쩔티비 어쩔티비' 마법 영창을 입력하면 '무자비'는 '어쩔티비'에 의해 지워지지만 '곰돌다운'은 지워지지 않아서 곰두리는 사망한다.
- 종강이야 - '곰돌다운'을 1회 막는 방어 마법을 씌운다. 이 마법은 중첩이 불가능하다.
  - '곰돌다운 곰돌다운 어쩔티비 어쩔티비'는 '어쩔티비'가 '곰돌다운' 하나만 지울 수 있어 곰두리가 사망한다, 두 번째 어쩔티비는 발동되지 않는다.
  - 하지만, '곰돌다운 곰돌다운 어쩔티비 종강이야'는 '어쩔티비'에 의해 '곰돌다운' 하나가 지워지고, 미리 씌워둔 '종강이야'덕분에 '곰돌다운'으로부터 살아남을 수 있다.

(다음 장에 계속)

- ‘곰돌다운 곰돌다운 종강이야 종강이야’의 경우, ‘종강이야’가 중첩이 불가능하기 때문에 ‘곰돌다운’을 한 번 밖에 막지 못해 곰두리가 사망한다.

입력

첫 번째 줄에 테스트케이스 개수를 입력받는다. 그리고 두 번째 줄 부터 마법 영창을 입력받는다.

마법 영창의 조건은 다음과 같다.
- 주문은 최대 8개까지 중첩이 가능하며, 8개가 넘는 주문이 영창되면 9번째 주문부터 효과가 발동되지 않는다.
- 주문은 스페이스로 구분한다.
- ‘곰돌다운’, ‘무자비’, ‘어쩔티비’, ‘종강이야’ 외의 주문을 입력하면 무조건 ‘마법역류’를 출력한다.
- 곰두리의 주문이 마법 영창에 포함되지 않을 수 있다.

입력예시는 다음과 같다.
3
곰돌다운 어쩔티비 종강이야 곰돌다운 무자비 어쩔티비 종강이야
곰돌다운 어쩔티비 곰돌다운 종강이야 곰돌다운 곰돌다운 어쩔티비 어쩔티비
곰돌다운 곰돌다운 곰돌다운 무자비 곰돌다운 어쩔티비

출력

마법 영창 로직을 해결해서 결과를 출력한다. 만약 곰두리가 생존했으면 ‘곰두리 생존’을 출력하고, 곰두리가 사망했으면 ‘곰두리 사망’을 출력한다. 마법 영창 조건에 위배된 주문을 입력하면 무조건 ‘마법역류’를 출력하고 다음 테스트케이스로 넘어간다.

입력예시의 경우 출력결과는 다음과 같다.
- 곰두리 생존
- 곰두리 사망
- 곰두리 생존

입력 예시의 경우 다음과 같이 로직이 발동된다.
- 종강이야 어쩔티비 무자비 곰돌다운 종강이야 어쩔티비 곰돌다운
- 어쩔티비 어쩔티비 곰돌다운 곰돌다운(방어실패) 종강이야 곰돌다운 어쩔티비 곰돌다운
- 어쩔티비 곰돌다운 무자비 곰돌다운 곰돌다운 곰돌다운

5. 회문수 (30점)

문제

회문은 앞으로 읽어도, 뒤로 읽어도 똑같은 문장을 의미한다. '소주 좀 주소', '적 명치에 치명적', '인싸 의사의 싸인'과 같이 앞에서 읽어도 똑같고 뒤에서 읽어도 똑같은 문장을 회문이라고 부를 수 있다. 그러면 회문의 뜻에 근거해서 '12321', '12344321'처럼 앞에서 읽어도 똑같고 뒤에서 읽어도 똑같은 수를 회문수라고 가정하자.

우리는 정수를 입력 받고 회문수인지 아닌지 구분하는 프로그램을 만들어 볼 것이다. 만약 입력받은 정수가 회문수라면 'Yes'라는 단어를 출력하면 된다. 하지만 입력받은 정수가 회문수가 아니라면 해당 숫자에서 가장 가까운 회문수를 찾아보자.

문제 이해를 위해 몇가지 예시를 들어보자. 20200202는 회문수가 맞으므로 'Yes'라는 단어를 출력하면 된다. 하지만 9998 같은 경우 회문수가 아니다. 이 경우 가장 가까운 회문수는 9998에서 +1된 9999일 것이다.

그렇다면 10000의 경우는 어떨까? 이 경우, 제시된 숫자로부터 -1 된 9999와 +1된 10001 모두 정답의 후보군이 될 수 있다. 이렇게 정답의 후보가 2개 이상인 경우, 입력받은 수보다 큰 정답만 출력하면 된다. 즉, 10000이 입력된 경우엔 10001이 정답이 된다.

입력

첫 번째 줄에 테스트케이스의 개수를 입력받는다. 그리고 두 번째 줄부터 테스트케이스 개수만큼 피검사수를 입력받는다.

여기서 피검사수의 조건은 다음과 같다.

- 모든 피검사수 x는 0 <= x <= 100,000,000을 충족한다.
- 위 조건을 충족하지 않은 피검사수는 검사하지 않는다.
- 한 자리수 정수 (e.g. 3, 5, 7)는 회문수로 취급한다.

입력예시는 다음과 같다.

```
3
12421
12399
1000
```

출력

피검사수가 회문수면 Yes를, 회문수가 아니라면 해당 피검사수에서 가장 가까운 회문수를 출력한다.

- Yes
- 12421
- 1001