Supreme Competition.

- Prepared by: Sagun Shakya (https://github.com/sagsshakya)
- MSc. Data Science
- GITAM Institute of Science, Visakhapatnam.
- Email: sags.shakya@gmail.com (mailto:sags.shakya@gmail.com)

Race is generally organized by distance but this race will be organized by time. In order to predict the winner we will check every 2 seconds.

Let's say total race time is 7 seconds we will check for (7-1) seconds. For 7 sec : We will check who is leading at 2 sec, 4 sec and 6 sec. Participant who is leading more number of times is winner from prediction perspective. Now our task is to predict a winner in this marathon. Note:

At particular time let say at 4th second, top two (top N, in general) participants are at same distance, then in this case both are leading we will increase count for both (all N). And after calculating at all time slices, if number of times someone is leading, is same for two or more participants, then one who come first in input sequence will be the winner. Ex: If participant 2 and 3 are both leading with same number, participant 2 will be the winner.

Constraints 1 <= T <= 100 1 <= N <= 100 Input Format First line contains a single integer N denoting the number of participants Second line contains a single integer T denoting the total time in seconds of this Marathon. Next N lines (for each participant) are as follows : We have T+1 integers separated by space. First T integers are as follow: ith integer denotes the number of steps taken by the participant at the ith second. T+1st integer denotes the Distance (in meters) of each step. Output Index of Marathon winner, where index starts with 1.

Example Input 1 3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4
Output
2
Explanation
3 (No. of candidate)
8 (Total time of Sprint (In seconds))
2 2 4 3 5 2 6 2 3 (data for 1st candidate. First 8 integers denote number of steps per second and last integer denotes distance covered in each step i.e. 3)
3 5 7 4 3 9 3 2 2 (similarly, 2nd candidate's data).
1 2 4 2 7 5 3 2 4 (similarly, 3rd candidate's data).
At time 2: Here 2nd marathoner is leading

12 (2*3*+2*3*)
16 (3*2*+*5*2)
12 (1*4*+2*4*)
At time 4 :Here also 2nd marathoner is leading

33 (23+23 +43+33)
38
36
At time 6 :Here 3rd marathoner is leading
54
62
84
At time 8: Here 3rd marathoner is leading
78
72
104
Output:
2
Since, 2nd marathoner and 3rd marathoner are tied, we use 2nd one because it came earlier in input sequence.

In [10]:
```
N = int(input())   # Total number of sprinters.
T = int(input())   # Total time of the race.

steps_per_sec = []
distance_per_step = []    # in metres.
for ii in range(N):
    temp_data = [int(jj) for jj in input().split()]
    assert(len(temp_data) == T + 1)
    steps_per_sec.append(temp_data[:-1])
    distance_per_step.append(temp_data[-1])
```

```
3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4
```

In [11]:
```
steps_per_sec
```

Out[11]:
```
[[2, 2, 4, 3, 5, 2, 6, 2], [3, 5, 7, 4, 3, 9, 3, 2], [1, 2, 4, 2, 7, 5, 3, 2]]
```

In [12]:
```
distance_per_step
```

Out[12]:
```
[3, 2, 4]
```

Time Slices.

In [5]:
```
time_slices = tuple(range(2, T+1, 2))
time_slices
```

Out[5]:
```
(2, 4, 6, 8)
```

Score Record.

- Keeps the streak of each sprinter for each time slice.

In [6]:
```
# Initialization.
scores_dict = dict()
for ii in range(1, N+1):
    scores_dict[ii] = 0
scores_dict
```

Out[6]:
```
{1: 0, 2: 0, 3: 0}
```

Main Calculation Part.

In [7]:
```
for time in time_slices:
    scores_list = []    # keeps track of the score for time_slices = time.
    for ii in range(N):
        score = (sum(steps_per_sec[ii][:time]) * distance_per_step[ii])   # Score
        scores_list.append(score)

    for sprinter, score in enumerate(scores_list, start = 1):
        if score == max(scores_list):
            scores_dict[sprinter] += 1     # This line handles the tied cases ver
```

Answer.

- To handle the case when two or more sprinters have the same scores at the end.
  - The situation is dealt by treating he/she, who had come up first in the input, as the winner.

In [8]:
```
for sprinter,score in scores_dict.items():
    if score == max(scores_dict.values()):
        print(sprinter)
        break
```

```
2
```

Joint Code.

In [9]:
```
N = int(input())   # Total number of sprinters.
T = int(input())   # Total time of the race.

steps_per_sec = []
distance_per_step = []    # in metres.
for ii in range(N):
    temp_data = [int(jj) for jj in input().split()]
    assert(len(temp_data) == T + 1)
    steps_per_sec.append(temp_data[:-1])
    distance_per_step.append(temp_data[-1])

### Time Slices.
time_slices = tuple(range(2, T+1, 2))

#### Score Record.
scores_dict = dict()
for ii in range(1, N+1):
    scores_dict[ii] = 0

#### Main Calculation.
for time in time_slices:
    scores_list = []
    for ii in range(N):
        score = (sum(steps_per_sec[ii][:time]) * distance_per_step[ii])   # Score
        scores_list.append(score)

    for sprinter, score in enumerate(scores_list, start = 1):
        if score == max(scores_list):
            scores_dict[sprinter] += 1


for sprinter,score in scores_dict.items():
    if score == max(scores_dict.values()):
        print(sprinter)
        break
```

```
3
8
2 2 4 3 5 2 6 2 3
3 5 7 4 3 9 3 2 2
1 2 4 2 7 5 3 2 4
2
```

The End.