Kitty

The **Kitty** is a very dangerous and strange creature. She loves to eat but also loves to collect souls. Not ordinary souls but the souls of coders. In order to not collect your soul Kitty assigned you a task to find how much food and souls she could collect in any iteration. Kitty doesn't know she can be deadlocked and the rest of the coders will be saved. There is a catch. Kitty could escape a deadlock if she leaves a soul or food in that cell.

Your task is to calculate the food and the souls collected by Kitty or to output that she is deadlocked.

On the first line of the input you will receive the positions of the **coder souls ("@")**, **food ("*")** and **deadlocks ("x")** as string.

On the second line of the input you will receive the path of the Kitty as string with integers **separated by single space**. Positive means, move to the right, negative means, move to the left.

The starting position will **always be at index 0**.

The final result is either the souls, food and deadlocks count or a string informing that the Kitty is deadlocked. The format is shown in the zero tests and the example.

@ - symbol for coder soul
***** - symbol for food
x - symbol for deadlock

Example

```
@@@xx@*
1 -1 -1 4
```

- We start at **position 0** and collect a soul **@**
- We move **1 time** to the **right** and now we are at **position 1** and collect a soul **@**
- We move **1 time** to the **left** and now we are at **position 0** and nothing is there because we already collected it, so we continue our path.

- We move **1 time** to the **left** and now we are at **position 6** (if you exit the array you should enter from the other side) and we collect food *****
- We move **4 times** to the **right** and now we are at **position 3** and there is a deadlock.
  - If the deadlock is on **even** position the Kitty should leave a **soul (@)** there in order to escape.
  - If the deadlock is on **odd** position the Kitty should leave **food (*)** there in order to escape.
- **Position 3** is **odd** so we leave **food (*)** to escape the deadlock.

At the end we have collected **2 souls** and **1 food** and we have left **1 food** to escape a deadlock. So the final result is:

```
Coder souls collected: 2
Food collected: 0
Deadlocks: 1
```

Input

All input data is read from the standard input (the console)

- On the first line you will receive the positions of souls, food and deadlocks as a string
- On the second line you will receive the path of the Kitty as string of integers separated by single space.

Output

The output data is printed on the standard output (the console)

- If the Kitty successfully escaped the deadlocks and finishes her path you should output on the console in the format
  - Where number is the final result of each type

```
Coder souls collected: [number]
Food collected: [number]
Deadlocks: [number]
```

- If the Kitty is deadlocked you should output on the console this line.
  - Where [number] is the count of the jumps before deadlock

  ```
  You are deadlocked, you greedy kitty!
  Jumps before deadlock: [number]
  ```

Constraints

- **The input data will always be correct and there is no need to check it explicitly**
- **Time limit: 0.05s**
- **Memory limit: 16MB**

Sample Tests

Sample Input 1

```
@@@xx@*
1 -1 -1 4
```

Sample Output 1

```
Coder souls collected: 2
Food collected: 0
Deadlocks: 1
```

Sample Input 2

```
x@*@*@*
2 -1 2 -1
```

Sample Output 2

```
You are deadlocked, you greedy kitty!
Jumps before deadlock: 0
```

Sample Input 3

```
@*@*@*xxx
1 -1 1 -1 2 1 1 1 1 1 1
```

Sample Output 3

```
Coder souls collected: 1
Food collected: 2
Deadlocks: 3
```