## Radioactive Mutant Vampire Bunnies

*You come across an old JS Basics teamwork game. It is about bunnies that multiply extremely fast. There's also a player that should escape from their lair. You like the game, so you decide to port it to Python because that's your language of choice. The last thing left is the algorithm that* determines *if the player will escape the lair or not.*

First, you will receive a line holding integers **N** and **M**, representing the lair's rows and columns.

Next, you receive **N** strings that can consist **only** of **"."**, **"B"**, **"P"**. They represent the initial state of the lair. There will be **only** one player. The **bunnies** are marked with **"B",** the **player** is marked with **"P"**, and **everything** else is free space, marked with a dot **"."**.

Then you will receive a string with **commands** (e.g., **LRRULUD**) - each letter represents the **next** **move** of the player:

* **L** - the player should move one position to the left
* **R** - the player should move one position to the right
* **U** - the player should move one position up
* **D** - the player should move one position down

**After** every step made, each bunny spreads **one position up**, **down**, **left**, and **right**. If the player **moves** to a bunny cell or a bunny **reaches** the player, the player dies. If the player goes **out** of the lair **without** encountering a bunny, the player wins.

When the player **dies** or **wins**, the game ends. All the activities for **this** turn continue (e.g., all the bunnies spread normally), but there are no more turns. There will be **no** **cases** where the moves of the player **end before he dies** **or** **escapes**.

In the end, **print** **the** **final state of the lair** with every row on a separate line. On the last line, print either **"dead: {row} {col}"** or **"won: {row} {col}"**. **"Row"** and **"col"** are the cell coordinates where the player has died or the last cell he has been in before escaping the lair.

### Input

* On the first line of input, the numbers **N** and **M** are received - the number of **rows** and **columns** in the lair
* On the following N lines, each row is received in the form of a string. The string will contain only **"."**, **"B"**, **"P"**. All strings will be the same length. There will be only one **"P"** for all the input
* On the last line, the directions are received in the form of a string, containing **"R"**, **"L"**, **"U"**, **"D"**

### Output

* On the first **N** lines, **print the final state of the bunny lair**
* On the last line, print:
  + If the player won - **"won: {row} {col}"**
  + If the player dies - **"dead: {row} {col}"**

### Constraints

* The dimensions of the lair are in the range **[3…20]**
* The directions string length is in the range **[1…20]**

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** | **Input** | **Output** |
| 5 6  .....P  ......  ...B..  ......  ......  ULDDDR | ......  ...B..  ..BBB.  ...B..  ......  won: 0 5 | 4 5  .....  .....  .B...  ...P.  LLLLLLLL | .B...  BBB..  BBBB.  BBB..  dead: 3 1 | 5 8  .......B  ...B....  ....B..B  ........  ..P.....  ULLL | BBBBBBBB  BBBBBBBB  BBBBBBBB  .BBBBBBB  ..BBBBBB  won: 3 0 |