## Radioactive Mutant Vampire Bunnies

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

First, you will receive a line holding integers **N** and **M**, which represent the rows and columns in the lair. Then you receive **N** strings that can consists **only** of **"."**, **"B"**, **"P"**. The **bunnies** are marked with "**B",** the **player** is marked with "**P**", and **everything** else is free space, marked with a dot **"."**. They represent the initial state of the lair. There will be **only** one player. Then you will receive a string with **commands** such as **LLRRUUDD** - where each letter represents the next **move** of the player (Left, Right, Up, Down).

**After** every step, each of the bunnies spread to the up, down, left and right (neighboring cells marked as **"."** **changes** their value to **"B"**). If the player **moves** to a bunny cell or a bunny **reaches** the player, the player has died. If the player goes **out** of the lair **without** encountering a bunny, the player has won.

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** stalemates where the moves of the player end before he dies or escapes.

Finally, 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 coordinates of the cell 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 next 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 the outcome - **"won:"** or **"dead:" + {row} {col}**

### Constraints

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

### Examples

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