[https://raw.githubusercontent.com/TelerikAcademy/Common/master/logos/telerik-header-logo.png](https://raw.githubusercontent.com/TelerikAcademy/Common/master/logos/telerik-header-logo.png)

***Telerik Academy Season 2016-2017 / JavaScript Fundamentals Exam - 05 June 2016***

**Task 2: Game of Trolls**

**Description**

**Wboup** and **Nbslbub** play an interesting game with **Lsjtujzbo** - a **Game of Trolls**. The idea of the game is the following:

* **Wboup** and **Nbslbub** are trolls.
* **Lsjtujzbo** is a princess.
* The trolls chase the princess on a field that has **R** rows and **C** columns. The field's coordinates are zero-based and are counted up to down, left to right - the first row is 0, the second 1, etc. The first column is 0, the second 1 and so on.
* **Wboup**, **Nbslbub** and **Lsjtujzbo** can move in 4 directions - up, down, left and right by **1** step.
* **Lsjtujzbo** is dangerous - she can lay traps. If a troll steps on a trap, it cannot move out of it, unless another troll comes to help him. If for an example **Nbslbub** falls in a trap, then he cannot move(even if there is such command) until **Wboup** stands in the same cell as well. When **Nbslbub** is rescued from the trap, the trap is destroyed and he can move freely again. **Lsjtujzbo** cannot fall into her own traps.
* The goal of the princess is to escape the field from the bottom right corner **(R - 1, C - 1)** or to capture both **Wboup** and **Nbslbub** in a trap and there is no troll in a cell around the princess.
* The goal of the trolls in to catch the princess - the princess is considered caught by the trolls if either **Wboup** or **Nbslbub** are standing in the same cell as the princess or in a neighboring cell. All 8 cells around the princess are considered neighboring.

**Input**

* The first element of the input array will be a string in the format R C, where **R** and **C** are the integer the dimensions of the field.
* The second element of the input will be a string in the format Rw Cw;Rn Cn;Rl Cl - pair of coordinates, separated by a **;**. Those are the starting coordinates of **Wboup**, **Nbslbub** and **Lsjtujzbo**.
* The rest if the input will contain commands as strings in the following formats:
  + "**mv unit dir**" - you should move **unit** (**Wboup**, **Nbslbub** or **Lsjtujzbo**) by **1** step in the direction **dir**.
  + "**lay trap**" - the princess lays a trap on her current location.

**Output**

* If one of the trolls catches the princess, output "**The trolls caught Lsjtujzbo at**", where **R** and **C** are the coordinates of the princess.
* If the princess escapes or both trolls fall into a trap, print "**Lsjtujzbo is saved! Rw Cw Rn Cn**", where **Rw Cw** are **Wboup**'s coordinates and **Rn Cn** are **Nbslbub**'s coordinates.

**Constraints**

* The directions for moving will always be one of the following: **u** for up, **d** for down, **l** for left, **r** for right.
* Traps do not stack on top of each other.
* The princess will never start next to the trolls. No two units will start on the same cell.
* The units cannot move out of the field. If a command will take a unit out of the field, don't execute it.
* **R** will always be between 5 and 30, inclusive.
* **C** will always be between 2 and 30, inclusive.
* Memory limit: **24MB**
* Time limit: **0.3s**

**Sample tests**

**Sample input 1**

[

'5 5',

'1 1;0 1;2 3',

'mv Lsjtujzbo d',

'lay trap',

'mv Lsjtujzbo d',

'mv Wboup r',

'mv Wboup r',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Wboup d',

'mv Wboup d'

]

**Sample output 1**

The trolls caught Lsjtujzbo at 4 3

**Sample input 2**

[

'7 7',

'0 1;0 2;3 3',

'mv Lsjtujzbo l',

'lay trap',

'mv Lsjtujzbo r',

'lay trap',

'mv Lsjtujzbo r',

'lay trap',

'mv Lsjtujzbo d',

'mv Lsjtujzbo r',

'mv Lsjtujzbo r',

'mv Lsjtujzbo r',

'mv Lsjtujzbo r',

'mv Wboup d',

'mv Wboup d',

'mv Wboup l',

'mv Wboup l',

'mv Nbslbub r',

'mv Nbslbub r',

'mv Nbslbub r',

'mv Nbslbub d',

'mv Lsjtujzbo d',

'mv Lsjtujzbo l',

'mv Lsjtujzbo l',

'mv Nbslbub l',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Wboup d',

'mv Wboup d',

'mv Wboup r',

'mv Lsjtujzbo d',

'mv Wboup d',

'mv Wboup d',

'mv Wboup r',

'mv Lsjtujzbo r',

'mv Lsjtujzbo r'

]

**Sample output 2**

Lsjtujzbo is saved! 6 2 3 4

**Sample input 3**

[

'8 8',

'1 3;0 3;5 5',

'mv Lsjtujzbo l',

'mv Lsjtujzbo l',

'lay trap',

'mv Lsjtujzbo r',

'lay trap',

'mv Lsjtujzbo r',

'lay trap',

'mv Lsjtujzbo d',

'lay trap',

'mv Lsjtujzbo d',

'lay trap',

'mv Wboup r',

'mv Wboup r',

'mv Wboup d',

'mv Wboup d',

'mv Wboup d',

'mv Wboup d',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Lsjtujzbo l',

'mv Nbslbub d',

'mv Nbslbub r',

'mv Nbslbub r',

'mv Nbslbub d',

'mv Nbslbub d',

'mv Nbslbub d'

]

**Sample output 3**

The trolls caught Lsjtujzbo at 7 4

**Solution**

function *solve*(*args*) {  
  
 Array.**prototype**.first = function () {  
 return this[0];  
 }  
  
 Array.**prototype**.last = function () {  
 return this[this.**length** - 1];  
 }  
  
 const sizes = *args*[0].split(**' '**).map(Number),  
 field = [],  
 starts = *args*[1].split(**';'**).map(*x* => *x*.split(**' '**).map(Number)),  
 units = {  
 **'Wboup'**: { **r**: starts[0][0], **c**: starts[0][1], **trapped**: false },  
 **'Nbslbub'**: { **r**: starts[1][0], **c**: starts[1][1], **trapped**: false },  
 **'Lsjtujzbo'**: { **r**: starts[2][0], **c**: starts[2][1] },  
 },  
 Wboup = units[**'Wboup'**],  
 Nbslbub = units[**'Nbslbub'**],  
 Princess = units[**'Lsjtujzbo'**],  
 deltas = {  
 **'l'**: { **r**: 0, **c**: -1 },  
 **'r'**: { **r**: 0, **c**: 1 },  
 **'u'**: { **r**: -1, **c**: 0 },  
 **'d'**: { **r**: 1, **c**: 0 },  
 };  
  
 for(let i = 0; i < sizes[0]; i += 1) {  
 field.push([]);  
 }  
  
 function *isInside*(*r*, *c*) {  
 return (0 <= *r*) && (*r* < sizes[0]) && (0 <= *c*) && (*c* < sizes[1]);  
 }  
  
 function *caught*() {  
 const byWboup = (**Math**.abs(Wboup.**r** - Princess.**r**) <= 1) && (**Math**.abs(Wboup.**c** - Princess.**c**) <= 1),  
 byNbslbub = (**Math**.abs(Nbslbub.**r** - Princess.**r**) <= 1) && (**Math**.abs(Nbslbub.**c** - Princess.**c**) <= 1);  
  
 return byWboup || byNbslbub;  
 }  
  
 function *escaped*() {  
 return (Wboup.**trapped** && Nbslbub.**trapped**) || ((Princess.**r** === (sizes[0] - 1)) && (Princess.**c** === (sizes[1] - 1)));  
 }  
  
 function *onSameCell*() {  
 return (Wboup.**r** === Nbslbub.**r**) && (Wboup.**c** === Nbslbub.**c**);  
 }  
  
 const commands = *args*.slice(2);  
  
 for(const cmd of commands) {  
  
 const split = cmd.split(**' '**);  
  
 if(split.first() === **'mv'**) {  
 let delta = deltas[split.last()],  
 unit = units[split[1]];  
  
 if(!unit.**trapped** && *isInside*(unit.**r** + delta.**r**, unit.**c** + delta.**c**)) {  
 unit.**r** += delta.**r**;  
 unit.**c** += delta.**c**;  
  
 if(field[unit.**r**][unit.**c**]) {  
 if(*onSameCell*()) {  
 Wboup.**trapped** = false;  
 Nbslbub.**trapped** = false;  
 field[unit.**r**][unit.**c**] = false;  
 } else if(unit !== Princess) {  
 unit.**trapped** = true;  
 }  
 }  
 }  
 } else if(cmd === **'lay trap'**) {  
 field[Princess.**r**][Princess.**c**] = true;  
 } else {  
 throw new Error(**'Your program sucks!'**);  
 }  
  
 if(*caught*()) {  
 **console**.log(**`The trolls caught Lsjtujzbo at** ${Princess.**r**} ${Princess.**c**}**`**);  
 return;  
 } else if(*escaped*()) {  
 **console**.log(**`Lsjtujzbo is saved!** ${Wboup.**r**} ${Wboup.**c**} ${Nbslbub.**r**} ${Nbslbub.**c**}**`**);  
 return;  
 }  
 }  
}