Introduction

You are required to implement a JavaScript function that is able to choose a *single* move in the game ShootingRange, given the current game state.

ShootingRange is a fairground style game where an AI Bot is moved side to side in a grid aiming to shoot targets that appear randomly within three alleyways. The game grid is split between an area where the AI bot can traverse, and the area where the targets appear.

The total game area is a grid of 5 x 4 cells, running from (0,0) in the top-left to (4,3) in the bottom right. The targets appear in cells (row: 0, col: 0), (row: 0, col: 2), and (row: 0, col: 4), and the player moves along (row: 3) and can move up into firing positions (row: 2, col: 0), (row: 2, col: 2) and (row: 2, col: 4).

When playing a local test game (explained more below) the board will be displayed like this:

```
       0 1 2 3 4

0   |1       3|
1   |_   _   _|
2   | |_| |_| |
3   |         X|
```

In this visualisation, cells are separated from one another by the "|" character. You can see the location of the player from the position of the X character. The locations of Targets are displayed using the number of moves they remain before disappearing (i.e. 5 means it will be there for 5 more moves, 2 means two more moves etc).

Targets will appear in the different alleyways randomly and will disappear after **9** turns or when they are successfully shot by the AI bot. The pattern to which they appear is randomised every tournament, but all AI bots play the same pattern as each other. The test game is **NOT** randomised, however.

In the game, points are awarded for shooting targets, and deducted slightly if an invalid move was made at any time in the game. Highest score wins!

Every turn, each bot can take *one* action, chosen by returning a single string from the function containing their code. The action the bot takes can be: "NORTH", "EAST", "SOUTH", "WEST", "FIRE" or "PASS".

NORTH, EAST, SOUTH and WEST cause the bot to attempt to move in the corresponding direction (Up, right, down, left).

FIRE will shoot a target if the bot is positioned in the firing position that shares the same column as the target. For example if there is a target in (row: 0, col: 0), the play must be in (row: 2, col: 0) for the shot to hit the target. (Note: Simply being in the correct column is not enough, the bot needs to be in the firing position (row: 2) too!).

Bots can choose to PASS if they do not want to take an action on the turn. This is considered a valid move (See scoring for further information).

The Function

You are given the function 'play(location, targetInformation, turnNumber)' to implement. You do not need to alter this function signature - just write your code inside this function.

Your function should always return one of the actions listed above, as a string. Returning any other value is invalid, as is returning an action which cannot be carried out in the current game state (ie, "EAST" when you are at the edge of the grid, or "FIRE" when you are not in a firing position corresponding to a target.)

Your function is passed three parameters (location, targetInformation, turnNumber), and will be called each time it is your bot's turn to play.

location (object) :

location is an object of the form:

```
{
    row: <number>
    col: <number>
}
```

It indicates the current position of your bot on the grid.

targetInformation (array) :

targetInformation is an array of objects, of the form:

```
{
    location: {
        row: <number>,
        col: <number>}
    },
    turnsUntilDisappeared: <number>
}
```

Each object in the array represents a target in the range, and has the location of the target, as well as the amount of turns remaining before the target will disappear.

In the event that there are no targets to display for the given turn, the targetInformation will be an empty array (have a length of zero).

turnNumber (number) :

The current turnNumber. This starts at 1, and will increment each time your function is invoked.

Scoring

Your bot will play in a tournament against the other team's bots. It will be scored based on performance.

- Hitting a target will earn you a point.
- Playing any invalid moves in the game marks that game as an invalid game, and will subtract 3 points from your score. **You have been warned**!

When you have a working bot, click "Submit Code" to send the current version to the server to compete in this tournament. You can submit your code as many times as you desire during the allowed time for the task.

The tournament will play your latest code, and scoring does not stack between tournaments so keep improving your code and your bots performance to win.

Local Test Games

You are able to test out your bot without submitting it by using the test game in the bottom-right of the development environment. Here you can see how your bot behaves in a game.

Notes:

- If your code throws an error, your move will be skipped.
- If your move is invalid, your move will be skipped.
- The game ends when there are no more targets remaining or scheduled to appear, or there have been 50 turns; whichever happens first.
- You are able to make as many submissions of your code as you want - each submission is stored for discussion later and we love to see how your solution progressed!
- When running a test game, calling console.log() with a string argument will cause the string you pass to appear as part of the output in the bottom right.
- Feel free to create as many other functions as you feel is helpful, just make sure you keep the signature of the main "play" function exactly as it is!

Good luck!