Function Name: `pokeStats`

Inputs:

1. (*double*) NxM array with missing stats
2. (double) NxM array with numbers needed to fill in missing stats in first input

Outputs:

1. (*double*) NxM filled-in array

Topics: (*array masking*)

Background:

After years and years of dedication and hard work you've finally done it; you've collected all the Pokemon and made Ash proud. Unfortunately your GameBoy decided to act up and replaced some of the stats with `-1`. Lucky for you, you always kept a backup of all the stats, and now is the time to use it!

Function Description:

Given an array with `-1` representing missing values, replace the `-1`s in the first input with the corresponding values in the second input.

Example:

```
missingStats = [34, -1                    replacementStats = [30, 50
                100, 5]                                       10, 5]
completeStats = pokeStats(missingStats, replacementStats)
completeStats → [34, 50
                 100, 5]
```

Notes:

- There can be multiple -1s in the first array.
- Both arrays are guaranteed to be the exact same size.

Function Name: `pokePics`

Inputs:

1. (*double*) 1xN vector of the encoded picture
2. (*double*) 1x2 vector containing the final dimensions of the picture

Outputs:

1. (*char*) The array of the decoded pokemon picture

Topics: (*array functions*), (*array math*)

Background:

You and your friend are trying to send each other pictures of your pokemon. Unfortunately, the tech of the poke world isn't what you are used to. As such the pictures arrive as numeric data. You have to use the one piece of tech you can rely on, MATLAB, to solve this problem!

Function Description:

To convert the input into an ASCII art picture, follow these steps:

1. Find the total number of elements in the input. Subtract this value from every element of the input array.
2. Reshape the vector into the new dimensions given by the second input.
3. Cast the array to type char, and then replace any uppercase letters with an asterisk (`'*'`) and any lowercase letters with a pound sign (`'#'`).

Example:

```
encoded = [46, 46, 46, 106, 99, 109, 46, 109, 103, 109, 46, 61, 46, 46]
pic = pokePics(encoded, [2, 7])

pic → ['  * *  '
       ' \___/ ']
```

Function Name: `escapeTeamRocket`

Inputs:

1. (*char*) An NxM array of the current map
2. (*char)* A character representing the initial direction of motion

Outputs:

1. (*char*) An NxM array of the final coursed map

Topics: (*conditionals*), (*iteration*)

Background:

After playing a few arcade games at the Rocket Game Corner in Celadon city, you stumble upon a mysterious button behind a poster, giving you access to Team Rocket's Hideout! Since you only brought Magikarp along with you, the Team Rocket grunts easily defeat you and throw you into one of their mazes! After trying to hours to escape, you resort to your trusty sidekick MATLAB (obviously programmed into your PokéWatch) to help you get out of this mess!

Function Description:

In this problem, you will be given an NxM character array map of the different tiles in the maze you need to escape along with an initial direction. In this map, you will initially be given up to 7 different characters:

- `'o'` represents your initial position
- `'X'` represents where your final position should be
- `'>'`, `'<'`,`'v'`, and `'^'` represent tiles that will change your direction of motion
    - `'>'` directs you right
    - `'<'` directs you left
    - `'v'` directs you down
    - `'^'` directs you up
- `'.'` is an empty tile with no special attributes

In the initial direction character, you will be given `'l'`,`'r'`,`'u'`, or `'d'`, which stand for left, right, up, or down respectively. This is the initial direction you should be moving in before hitting any change of direction tiles.

Lastly, at every location you've been at, replace the tile you were standing on with a `'#'` to represent the path you must travel along.

(*continued…*)

Example:

```
>> map = ['>.>X.';
          '..v.o';
          '.v..<';
          '^.<..'];
>> dir = 'l';
>> key = escapeTeamRocket(map,dir);

   key → ['###X.';
          '#.###';
          '#v#.<';
          '###..'];
```

Notes:

- Do not replace the final location with a '#' but do replace the initial location with a '#'
- You will never cross over the same path previously traveled or travel outside of the matrix dimensions
- Keep in mind that up and down correspond to changing rows, while left and right correspond to changing columns.

Function Name: `pokeMigration`

Inputs:

1. (*double*) An MxN array representing the likelihood a pokemon is in that region

Outputs:

1. (*double*) An M-2xN-2 array representing the updated likelihood a pokemon is in that region

Topics: (*nested iteration*), (*array indexing*)

Background:

You are competing against Gary to fill up your respective pokedexes, but you've been struggling to find new pokemon. As a member of the Georgia Tech gym, you've been learning how to use MATLAB to become a better trainer. You decide to use MATLAB to track some of those pokemon that have been eluding you and your pokedex.

Function Description:

You will be given an array of numbers representing the likelihood that a pokemon can be found in a certain region. Return a new array where each element is the average of the value from the old array and its eight surrounding neighbors. Since edges and corners do not have eight neighbors, they are deleted from the new array. A trivial example is as follows:

```
[ 2 4 2;
  4 3 4;  →  [3]
  2 4 2]
```

Example:

```
>> oldMap = [4 5 3 10;
             5 0 1 7;
             5 6 9 11;
             9 8 2 22];
>> updatedMap = pokeMigration(oldMap)
>> updatedMap → [4.22 5.78;
                 5.00 7.33]
```

Notes:

- Round the updated array to 2 decimal places.
- The initial array is guaranteed to be at least 3x3

Function Name: `swordAndShield`

Inputs:

1. (*double*) An Nx7 array showing a Pokemon's data (Pokedex No and 6 stats)
2. (*double*) The column index of the stat that you will use to sort your array
3. (*double*) The minimum value for that stat needed to qualify for the new game

Outputs:

1. (*double*) An Mx7 array of all the qualifying Pokemons' data

Banned Functions: `sortrows`

Topics: (*masking*), (*multiple sorts*)

Background:

You are creating a new pokemon game with your CS1371 pals! However, you are lazy and don't want to include all 890 Pokemon in the new game. Instead, you've decided to only include the pokemon that have cool battle stats.

Function Description:

Your MATLAB function will first eliminate all pokemon that do not have the minimum value for a specific stat whose column number is indicated by the second input. It will then sort the array by the chosen column index in *descending* order. In the case of a tie, order the ties in descending Pokedex number. Finally, output the sorted array.

Hint:

- MATLAB's sort function places tied values in their original order.

(*continued…*)

Example:

Pokedex No	HP	Attack	Defense	Sp. Attack	Sp. Defense	Speed

```
>> original =
     [1     0          1          0          1          0          0;
      6     0          0          0          3          0          0;
      60    0          0          1          0          0          2;
      212   0          2          0          0          0          0;
      435   2          0          0          2          0          0;
      518   4          3          0          3          2          0;
      792   0          0          0          3          0          0];

>> genEight = swordAndShield(original, 5, 2)
   genEight →
     [792   0          0          0          3          0          0;
      518   4          3          0          3          2          0;
      6     0          0          0          3          0          0;
      435   2          0          0          2          0          0];
```