Notice

For this week's homework, some problems can only be tested after 'loading' certain `.mat` files as instructed in the `hw05.m` file. You can load these files two ways before running test cases: right click the `.mat` file in your directory and select 'Load' OR use the built-in `load()` function (see function documentation: https://www.mathworks.com/help/matlab/ref/load.html). If you are writing a test script, it would be wise to use the `load()` function appropriately. **DO NOT** use the `load()` function in your function code! To test your function(s), load the `.mat` file first and then run the function with the specified inputs just as you have been doing up until now. The Autograder will automatically load the appropriate `.mat` files for you before grading your homework. If `load()` is included in your function, it will error, and you will not receive credit.

Happy coding!
~Homework Team

Function Name: `arrReplace`

Inputs:

1. (*double*) An MxN array
2. (*double*) Another MxN array
3. (*double*) A number

Outputs:

1. (*double*) The first original MxN array with replacements made

Function Description:

Write a function called arrReplace() that inputs two arrays and a number; it should replace all instances of that number in the first array with the values in the corresponding positions of the second array. If the number does not exist in the array, it should just return the original array with no changes.

Example:

```
arr1 =  [11,  8;          arr2 =  [5,   2;          num = 8
          8,  4]                   7,   6]

>> result = arrReplace(arr1, arr2, num)
result =   [11,  2;
            7,  4]
```

Notes:

- The two input arrays are guaranteed to have the same dimensions.

Function Name: `decodeASCII`

Inputs:

1. (*double*) An NxM array of ASCII values
2. (*double*) A vertical scaling factor
3. (*double*) A horizontal scaling factor
4. (*char*) Border character
5. (*double*) Border width

Outputs:

1. (*char*) An array with the decoded ASCII image

Function Description:

Who ever said engineering students can't appreciate fine art? In this problem, you will work with ASCII art, or images made from an array of characters. Given an array of ASCII codes, you will convert it to characters and rearrange it to decode the hidden picture.

To decode the picture, use the following process:

1. Convert the array to the characters represented by the ASCII values.
2. Flip the upper right quadrant vertically.
3. Switch the two lower quadrants.
4. Use `linspace()` to resize the image by the vertical and horizontal scaling factors (see resize equation below). This means certain array elements will be repeated to resize to the desired dimension.
5. Apply a border to the image with the specified width and the border character.

Resize Vector Equation:

```
idx = floor(linspace(1, length(vec) + (frac - 1)/frac, length(vec)*frac));
vec = vec(idx);
```

Where `frac` is the fraction you want to stretch/compres your vector. Extend this equation to stretching/compressing arrays.

Notes:

- Input arrays will always have an even number of rows and columns.

Function Name: `GSquare`

Inputs:

1. (*double*) A 1x(2*N + 1) vector of homework grades
2. (*double*) A 1xN vector of comment grades
3. (*char*) An Nx3 character array representing the grade mode

Outputs:

1. (*double*) The final homework grade

Background:

As part of an assignment to work on your WOVEN skills for English class, you are assigned a pen-pal in a foreign country who you correspond with through email and vlogs. It just so happens that your pen-pal happens to attend the best engineering school in Georgia, and is also currently taking a CS class! At the end of the semester, he informs you via vlog that his homework grades for the class are not reported correctly in his online gradebook since the software, GSquare, is incapable of taking into account whether a homework assignment is the maximum or average of his original and resubmission. Since he tells you that he too cannot do this simple math, and that his TAs told him at the beginning of the semester not to ask what his real homework grade should be, you offer to write a function that will do it for him.

Function Description:

The first input represents your homework grades, alternating between the respective assignment's original grade and resubmission grade. The rest of the elements follow this alternating pattern except for the final entry, which is the grade on the extra credit assignment.

```
in1 = [100 0 95 0 75 95  ...  88]
       HW1   HW2   HW3       EC
```

The second input represents your comment grades, which can add up to 10 additional points per homework. So a 95 comment grade adds 9.5 points.

```
in2 = [ 100    0    95   ... ]
       HW1   HW2   HW3
```

The third input will be a character array with rows reading either `'avg'` or `'max'`, which will determine whether you should take the average or maximum of the original and resubmission grades for the corresponding homework.

```
in3 = 'avg
       max
       avg
        ... '
```

Use this information to calculate the overall homework grade. Remember that a resubmission can never hurt your grade, only improve it: if the resubmission grade is lower, or if the assignment was not re-submitted (represented by a 0), then the resubmission grade should

be ignored. The extra credit assignment does not have a resubmission or commenting grades. The EC assignment replaces the student's lowest homework grade. If the EC assignment is the lowest grade, then do not count the extra credit. The final homework grade should be calculated as the mean of the grades of each assignment, after accounting for the grade mode, comments, and extra credit.

Notes:

- A student will have grades for each of their assignments, along with the extra credit assignment, ranging from 0 to 100.
- The string in the first row of the character array corresponds to the first hw assignment, or the first two elements of the homework grade vector. The second row to the next two, and so on.
- Comment grades will be out of 100 in the input vector.
- The grade mode and comment grades should both be incorporated before replacing the lowest grade with the extra credit grade.
- If two or more grades are tied for the lowest grade, replace only one grade.
- Round your answer to the first decimal place.

Function Name: `matCraft`

Inputs:

1. *(double)* An integer array of the elevation map of the world
2. *(double)* A positive integer array of the rating map of the highest blocks in the world
3. *(double)* 1x2 vector containing your current location

Outputs:

1. *(double)* A vector containing the elevation and terrain scores
2. *(double)* The distance to the closest source of water

Banned Functions:

`geomean()`

Function Description:

After finishing last week's CS1371 homework, you decide to relax by playing the hottest new sandbox game, matCraft! matCraft is an epic game of survival where you can build amazing things, like a computer that can run MATLAB. You have created several worlds in matCraft and wish to find out which world is the best. To make things easier, you decide to write a function in MATLAB that evaluates how good each world is!

Each world is characterized by two maps. The first map is an array of doubles which represents the elevation at any point in the world in terms of the number of blocks above sea level. Non-positive values mean that the location is underwater. The depth of the water in number of blocks is equivalent to the magnitude of the negative value in the array. The second map is an array of doubles which represents the composition rating of the highest block at that point in the world, ranging from 1 (dirt) to the highest number in the array (diamond).

Given these two maps and a vector containing your row and column coordinates, in that order, write a function that outputs several pieces of information about the world. The first output should be a vector containing the elevation score and terrain score, in that order. The elevation score can be calculated by taking the sum of the elevations of the mountainous terrain (where elevation is greater than or equal to 11) and adding it to the sum of the depths of the underwater terrain (where elevation is less than or equal to 0). The terrain score is calculated by finding the geometric mean (refer to notes) of the top block ratings for the points that are neither mountainous nor underwater. The function's second output should be the Cartesian distance between your current location and the nearest underwater location.

If, for example, your row and column coordinates were `[1,3]`, you would be located where both maps below have a value of `5`, and the distance to the closest body of water would be `1.414`.

```
eleMap =                        blockMap =
       2     2     5                   1     3     5
      -1    -2    11                   1     1     2
      -1     1     6                   1     2     3
```

Notes:

- It is guaranteed that there will be at least one underwater location on the map.
- When calculating the elevation score, the sum of the underwater regions should be a negative number, since you are summing negative values.
- The rating for diamond will always be the greatest value in the second array, but can vary between test cases.
- You can use `randi()` to generate sample maps to test your code. Be sure to include both positive and negative numbers in your elevation map and only positive numbers in your block map.
- Do not use large maps when testing your code, otherwise you may wind up with a number too large for MATLAB to handle when calculating the geometric mean. A good cap on the size of a map is 30x30.
- Round each of the scores and the distance to the third decimal place.
- The geometric mean of n terms x_1, x_2, … x_n is defined as: $\bar{x} = \sqrt[n]{x_1 x_2 \dots x_n}$

Hints:

- The `nthroot()` and `prod()` functions will be useful in calculating the geometric mean.
- Both the first and second outputs `find()` may be useful in calculating the distances.
- The method you used to solve `cartDist()` for Homework 1 should be used to calculate the distances in this problem.

EXTRA CREDIT

Function Name: `sieve`

Inputs:

1. *(double)* An upper limit, n

Outputs:

1. *(double)* A row vector of prime numbers

Banned Functions:

`primes(), isPrime(), ismember(), setdiff(), factor()`

Banned Coding Constructs:

Conditionals (if and switch statements), Iteration (for and while loops)

Function Description:

You are working at the great Library of Alexandria. It is the year 190 B.C. Your best buddy and chief librarian, Eratosthenes of Cyrene, comes to you with a problem. He thinks he has come up with a way to list all of the prime numbers less than or equal to a given number n! He is not sure how well it will work, though, so he asks you to implement it in MATLAB ancient Greek style. He tells you to check out this article for some very cool instructions and visualizations.

The idea is to start with all of the numbers from 1 to n, and eliminate multiples of 2, then multiples of 3, then multiples of 4, etc. until only prime numbers remain. However, Eratosthenes noticed that you only have to check for divisibility by numbers up to `floor(sqrt(n))`.

Your function should behave exactly the same way as the built in function `primes()`, but of course this function is banned!

If you know about conditionals and/or iteration from past coding experience, it may be tempting to try to use it for this problem. However, there is a way to solve the problem using only simple math, arrays, and indexing.

Notes:

- The input will always be a positive integer
- 1 is not prime!
- If the input number is prime, include it in your output.
- Use of iteration or conditionals to solve this problem will receive **no credit**

Hints:

- Use the `.*` and `./` operators to your advantage and remember that dimensions must match when multiplying matrices.
- Experiment with how `any()`, `all()` and `sum()` work on arrays (check out the 2nd input).
- Try to come up with some math on an array that will allow you to write a logical condition (or conditions) (read: masking!) to identify primes in the array.