Function Name: `bossFinder`

Inputs:

1. (*char*) A MxN character array of names

Outputs:

1. (*char*) The name of the boss

Topics: (*arrays*), (*indexing*)

Background:

It's time...You have been working like crazy and feel confident that you deserve a promotion. The one issue is that your company never actually told you who the boss is. However, luckily the CEO Blake is a huge MATLAB fan so he always would choose bosses by who had the best name in terms of the ascii value. You now know what you must do, and you pull up MATLAB to go get that promotion.

Function Description:

Given a character array of names of all the higher ups in the company, you must find who the boss is. The CEO determines the best name based on which names have the highest sum of their corresponding ascii values. You must calculate this value for the names and then index out the name of your boss.

Example:

```
array = ['emma';
         'john';
         'jack';
         'jane';
         'fred']
boss = bossFinder(array)
boss = 'john'
```

Notes:

- The name must be outputted exactly how it appears in the array
- Each of the names within the array will always be the same length.

Notes:

- The second input of the sum() function may help here

Function Name: `numberWorker`

Inputs:

1. (*double*) Mx4 array of loan amounts (principal)
2. (*double*) Mx4 array of interest rates (percent)
3. (*double*) Mx4 array of duration of loans (number of years)
4. (*double*) Amount representing interest goal

Outputs:

1. (*double*) 1x4 vector representing total expected interest from loans in each quarter
2. (*char*) Sentence describing how many quarters goal was made

Topics: (*array math operations*), (*formatted strings*), *(masking)*

Background:

You have landed a big job at Bank of Buzz after graduating from Georgia Tech. In an effort to impress your boss (and maybe get a raise), you decide to use your MATLAB skills to help loan profit analysis on the loans that the bank issued this year.

Function Description:

You are given 3 arrays, each column representing one of the 4 quarters in a business year. The first array (input 1) contains records of all of the loans that the bank gave out in a year (`P` in the formula). The second array (input 2) contains the interest rates for the corresponding loans given in percentages (`r` in the formula). Finally, the third array (input 3) contains the time that the loan was out for (`t` in the formula). Using the simple interest formula shown below, calculate the interest that the bank earns per loan (store this output in an array).

$$I = Prt$$

After you have calculated the interest, produce a sum for each column of the interest array. This gives you the total interest per quarter vector (output 1), round this value to 2 decimal
places. Count how many quarters the bank met or exceeded their quarterly interest goal, calculate the total interest for the entire year (round to the nearest cent) and output a string with the following format (output 2):

```
'The bank reached their goal <# of quarters> quarters out of the
 year. The total expected interest is $<yearly interest (with 2
                        decimals)>.'
```

Example:

```
loans = [2000  5000   100   6600;
         1050  3250  5100 100000;
          500 30000     0  25000]
```

```
intRates = [  7.5 12.5 50 5;
            16.6   10 14 4;
              40    5  0 2]

 time = [5 10  1 30;
         5 10 15 20;
         3 30  0 10]
goal = 10,000

[expIntQuarterly,    bankStatement]    =    numberWorker(loans,
intRates,
                                                  time, goal)

expIntQuarterly = [2221, 54500, 10760, 94900]
bankStatement = 'The bank reached their goal 3 quarters out of
the year. The total expected profit is $162381.50.'
```

Notes:

- Interest rates are given in percent form, so you must convert to a decimal before plugging in the formula.
- When displaying your yearly output in the sentence, ensure it is not in scientific notation and only 2 values are displayed after the decimal. (ex. $162381.50 rather than $162381.50 or 1.6238e+05)
- The yearly profit MUST be displayed as it is in the example, do not display in scientific notation or with any more than 2 decimal places.
- Your first output will likely display in scientific notation, this is okay.
- Remember, P = principal amount, r = interest rate, t = time (in years), I = interest
- Make sure to calculate the yearly total by summing the rounded quarterly totals.

Hints:

- If your string output looks off, remember that %d is not always the right choice for numeric values in the sprintf() function

Function Name: `graphcmp`

Inputs:

1. (*char*) A MxN char array representing a bar graph
2. (*char*) A KxL char array representing a misformated bar graph

Outputs:

1. (*char*) A formatted string describing the mistakes in the second graph

Topics: (*masking*), (*array operations*)

Background:

You are a hardworking intern at CS1371 Corp and have compiled some performance data on different employees to help your boss decide who should get a raise at the end of the quarter. However, your boss can be a bit of a dunce and has messed up your nicely formatted graphs! You must now find the differences between the two graphs and inform your boss of how they messed up!

Function Description:

You are given two char arrays that represent bar graphs. The first array contains your correctly formatted bar graph. In your array, each EVEN column will represent a bar on your graph. The ODD columns contain only white space (`' '`) to help your graph be more visually appealing. Starting from the bottom of the array, the bars are made of an arbitrary character that is **NOT** a space (`' '`). The second array is messed up and may have any number of the following errors: (1) the bars themselves may consist of a different character, (2) the bar heights may be different, (3) the second array may have been reshaped so the array dimensions are incorrect and the graph is unreadable.

You must determine how many bars in the second bar graph are incorrect. First, you should ensure your second array has the same dimensions as the first array. Second, you should compare the heights of each bar between the two graphs to count the number of bars that are different. Output the number of bars with incorrect heights in a string that follows the format: `'My boss' graph has <numIncorrectBars> bars that need fixed!'` where `numIncorrectBars` is the number of bars in the second graph that are of a different height than the first graph.

Example:

```
myGraph =
    [' # # # # '
     ' # # # # '
     ' # # # # '
     ' # # # # '];
```

```
bossGraph =
    [' %     '
     ' %   %'
     '      '
     '   %  '
     '   %  '
     '   %  '];
```

```
str = graphcmp(myGraph, bossGraph);
str → 'My boss' graph has 3 bars that need fixed!'
```

Notes:

- The even columns of myGraph have been highlighted for visual clarity, this is not a part of the problem.
- Every column with an even index in the graph counts as a bar, even if that bar's value is zero.
- The first and last column of the graph will always contain only spaces.
- You are guaranteed to have at least 1 bar in every graph
- You will always have the same number of bars for both graphs
- A bar of correct height but incorrect character does not have to be fixed and is not to be counted in the output

Hints:

- The `reshape()` function may be useful
- Masking might help you figure out the size of each bar
- When including an apostrophe in a string, use `''` (2 apostrophes) to tell MATLAB that the apostrophe is not supposed to be the end of the string

Function Name: `coinCounter`

Inputs:

1. (*double*) an N x 4 array containing varying amounts of pennies, nickels, dimes, and quarters in each column
2. (*char*) a 1x4 char vector detailing the type of coin in each of the columns

Outputs:

1. (*double*) a 5 x N array containing the coin amounts from the original array as well as each employee's total earnings in tips
2. (*char*) a sentence indicating the highest paid employee in tips and how much they earned for that given day

Topics: (*array masking*), (*transpose*), (*array manipulation*), (*sorting*), (*sprintf*)

Background:

It is the end of the day after a long shift bussing tables at Rays NY Pizza in Tech Square. You and your coworkers noticed that a ton of Georgia Tech students have been tipping in coins recently, so you decide to put your heads together and calculate who earned the most in tips that night. Everyone split their own coins into different piles of the types of coins they earned and counted how many of each type they had. To make it easier, you decide to use your MATLAB skills to calculate the amount each employee earned.

Function Description:

Given an N x 4 array (input 1) where each row of the array represents a different employee and each column of the array represents a certain type of coin, determine how much money in tips each of the employees made for a given shift. The values in the array represent the number of each type of coin they received. In the 1 x 4 char vector (input 2), 'p' represents pennies ($0.01), 'n' represents nickels ($0.05), 'd' represents dimes ($0.10), and 'q' represents quarters ($0.25). The order of the char vector represents the type of coin (denoted by the letters 'p', 'n', 'd', and 'q') and its corresponding column (based on its index) in the larger array. For example, the string 'pndq' would mean pennies are in the first column, nickels in the second, dimes in the third, and quarters in the fourth. Calculate each employees' earnings in tips and append a column to the right of the original array containing their corresponding totals.

Next, sort the entire array based on each employee's total amount of tips in descending order (the highest paid employee would be the first row, lowest paid employee would be the last). Finally, flip the rows and columns of the array, such that the tip totals are now found in the last row of the array and the employees and the corresponding number of coins they each have are represented by each column, and round all values to the second decimal place. Store the new rounded array in the first output.

Finally, output the following statement:

```
'Employee number <original employee row> was the highest paid
                and made $<amount> in tips today.'
```

Where the `original employee row` corresponds to the row in the original array (prior to sorting) where the highest-paid employee was found and `amount` corresponds to the amount in tips they made.

Example:

```
coinArr=  [5     3     8     2;
           6     3     2     0;
           4     2    11     9]
coinOrder= 'dqpn'

[tipArr, sentence]= coinCounter(coinArr, coinOrder)


tipArr →  [4       5       6;
           2       3       3;
           11      8       2;
           9       2       0;
           1.46    1.43    1.37 ]

sentence → 'Employee number 3 was the highest paid and made $1.46
             in tips today.'
```

Note: the highlighted row/column denotes employee #3

Notes:

- The coin array will contain only positive integer values
- The only chars that will be found in the second input are 'p', 'n', 'q', and 'd', appearing exactly as given above and occurring exactly one time each
- The coins can be given in any order
- When displaying your output sentence, ensure the dollar amount only displays two values after the decimal (ex. $5.98 rather than $5.9800)
- Be sure to round the final array as stated above, as to not cause a floating point error (slight rounding inaccuracies with decimals can cause the entire output to be considered incorrect)

Hints:

- Remember **conditionals and iteration are banned**
    - Think of a way you can do this problem without using conditionals. You are guaranteed to only have 4 types of coins and you know how much each coin is worth.
- Try using sprintf with %f

- Which operation would allow you to switch the rows and the columns?

Function Name: `timeUnscrambler`

Inputs:

1. (*double*) MxN array representing your scrambled time card

Outputs:

1. (*double*) Mx(N+1) array representing your fixed time card

Topics: (*array masking*), (*sorting arrays*), (*linear indexing*)

Background:

You've just been hired as a TA for Georgia Tech's newest computer science class, CS1731. In hopes that the CS department will give you a raise, you decide to take on some extra hours by working the overnight shift at the help desk. Each night, you log the hours you've worked on a timecard, and at the end of the week you turn it in to your boss in hopes of that prized promotion. One evening, however, you fall asleep for 5 minutes on the job and when you wake up, your pesky co-TA's have messed with your time cards to ruin your chances of impressing your boss! Desperate, you turn to MATLAB to help you unscramble the mess.

Function Description:

You are given an array of doubles representing your scrambled time card. Apply the following operations to unscramble your card:

1. Replace all negative numbers in your array with their positive equivalent (i.e. -5 → 5)
2. Flip the order of the elements in your array's major diagonal, for example:

```
[ 1  2  3  4          [ 6  2  3  4
  5  6  7  8     →      5  1  7  8
  9  0  1  2            9  0  6  2
  3  4  5  6 ]          3  4  5  1 ]
```

3. Sort the rows of your array in ascending order based on the first column (i.e. the numbers should get larger as you go down the rows in the first column).
4. The first column of your sorted array will represent the amount of time you have worked, in minutes. Replace this column with two separate columns in the final output, where the first column is the number of hours you have worked, and the second is the number of additional minutes. For example:

```
[ 100                  [ 1  40
  234            →       3  54
  52                     0  52
  300  ]                 5  0  ]
```

Example:

```
scrambled = [ -11    -23     3    -40
               8     382    -22     0
              -80    -49    123    -2
              563     4      4     63 ]

unscrambled = timeUnscrambler(scrambled)

unscrambled = [ 0    8   123    22   0
                1    3    23     3   40
                1   20    49   382   2
                9   23     4     4   11  ]
```

Hints:

- To linearly index out the values of the major diagonal, start by finding the starting index, step size, and stopping index for the diagonal first.
- Try writing out 5x3, 3x3, and 3x5 arrays and ask yourself the following questions:
    - How do the indices for the major diagonal behave?
    - What patterns do you observe (either in step sizes or stopping index values) in relation to the array's dimensions?
- How can you flip or reverse the order of a vector using linear indexing?