Name: ____

GTID (ex: 903XXXXXX): ____

CS 1371
Exam I
September 25th, 2019
Form A

- Write your name and GTID on every page or **you could lose 5 points on the exam.**
- Check the board at the front of the room for any updates/clarifications during the exam.
- All logical values should be denoted using `true` or `false` NOT 1 or 0
- An appendix containing useful information is provided separately.
- If there is a problem that you cannot figure out, skip it and move on. Be mindful of how many points each question is worth.
- If you run out of room, use the back page. Be sure to mark that you are continuing your answer on the back page!
- You have fifty minutes. Good Luck!

I pledge that I have neither given nor received help on this exam:

____ ____________________
Signature Date

Problem 1. Answer the following conceptual questions. (21 Points)

a. Determine whether the following function headers will be accepted by MATLAB without error. For EACH header fill one bubble for valid or invalid. **If the function header is invalid, rewrite the CORRECTED function header using the same function and variable names.** (3 points each).

```
function pink = mixColors(white, red)
```

◯ Valid ◯Invalid ____

```
function SchoolSpirit(oldGold, white)
```

◯ Valid ◯Invalid ____

```
Function [brown] = primaries(red, blue, yellow)
```

◯ Valid ◯Invalid `function [brown] = primaries(red, blue, yellow)`

b. The following function is defined in the current directory. Consider the function and answer the questions below.

```
function [fun] = musicMidtown(day, time)
   crowded = time ./ 2;
   rating1 = stage1(day, time);
   rating2 = stage2(crowded);
   factor = crowded + day;
   fun = (rating1 + rating2) .* factor;
end

function [crowded] = stage1(day, time)
   factor = 8 - day;
   crowded = time .* factor;
end

function [factor] = stage2(fun)
   crowded = fun .* 2;
   factor = crowded - str2num('1');
end
```

The following code is run in the Command Window.

```
>> num = 6;
>> crowded = mod(1, 4);
>> fun = musicMidtown(num, 2)
```

Name: ____

GTID (ex: 903XXXXXX): ____

Complete the table below with the values of each variable in the listed workspaces. If the variable does not exist in the workspace, write DNE. (1 point each).

	Command Window	`musicMidtown`	`stage1`	`stage2`
`fun`	35	35	DNE	1
`crowded`	1	1	4	2
`factor`	DNE	7	2	1

Problem 2. Answer the following tracing questions. (24 points)

```
1  function [out1, out2] = zoo(animals, price, quantity)
2    mask1 = (animals >= 'a' & animals <= 'z') | animals == ' ';
3    animals(~mask1) = [];
4    price = price .* quantity;
5    [num, idx] = max(price);
6    out1 = animals(1:idx+2);
7    mask2 = price <= num./2;
8    mask3 = quantity < num./2;
9    vec = mask2 & mask3;
10   out2 = price(vec);
11 end
```

Assuming the above function is defined in the current directory and the following code is written and run in the Command Window, answer the questions that follow:

```
>> animals = 'ap2e. do2g _c4at..'
>> price = [2 1 3]
>> quantity = [4 3 1]
>> [out1, out2] = zoo(animals, price, quantity)
```

a. What is the value stored in `out1` and `out2` in the Command Window? (4 points each)

`out1`: 'ape'

`out2`: [3 3]

b. If Line 2 is changed to the following:

```
mask1 = animals >= 'a' | animals <= 'z' | animals == ' ';
```

Will `mask1` change? If so, describe how it changes. (6 points)

Yes, the mask will change. Changing the & to an | means that every value in the mask will be true, as every single possible character is greater than/equal to 'a' OR less than/equal to z.

Name: ____

GTID (ex: 903XXXXXX): ______________________________________

c. If Line 4 is changed to the following, will the code error? If so, very briefly describe the error. If not, explain why an error does not occur. (5 points)

```
price = price(1:end-1) .* quantity;
```

```
Error, dimension mismatch
```

d. If Line 10 is changed to the following line, the function will not error. Will the value of `out2` change? Explain why it does or does not change, and if it does, provide the new value of `out2`. (5 points)

```
out2 = price(find(vec));
```

out2 will not change. The find() function returns all non-zero values, so when used on a logical vector, returns the numerical indices of all true values. The resulting vector of class double will index price at the exact same locations as the logical vector, so the output does not change.

Problem 3. Write scripts (short sequences of MATLAB code, not functions) that perform the following tasks. Do NOT hard code: use the given variable names, do not substitute their values. (25 points)

a. Given a character vector called `letters` and a char array `arr`, replace the underscores (`'_'`) in `arr` with the characters stored in `letters`. Assign the resulting sentence in a variable `sent` as a vector, as shown below in the example case. Note that once the array becomes a vector, the first row comes first, then the second, and so on. It is guaranteed that the number of underscores in `arr` is equivalent to the length of `letters`. (9 points)
Note that in the example case, there is a space after the `'w'` in the first row.
HINT: You may want to turn the array into a vector before replacing the underscores.

Example:
```
>> letters = 'hr?'
>> arr = ['Hi, _ow ';
          'a_e you_']

>> sent = ['Hi, how are you?']
```

Write your SCRIPT here:

```
arr = arr'
arr = arr( : )
sent = arr'
mask  = sent == '_'
sent(mask) = letters
```

Name: ____

GTID (ex: 903XXXXXX): ______________________________________

b. Given a `1xN` double vector stored in `nums`, find the number of even integers in the vector and store it in `evens`. Then delete all values of `nums` that are not divisible by the value stored in `evens`. (8 points)

Example:
```
>> nums = [4 12 10 15 8 20]

>> evens = 5
>> nums = [10 15 20]
```

Write your SCRIPT here:

```
evens = sum(mod(nums, 2) == 0)
mask = mod(nums, evens) == 0
nums(~mask) = []
```

c. Given the 1xN double vector `vec`, find the sum of the first half of `vec` and the product of the second half. Find the greater of the two values and store it in `great`. If the **sum** is greater than or equal to the product, output `true`. If the **product** is greater than the sum, output `false`. Store the logical in `out`. **Note**: `vec` is guaranteed to be an even length. (8 points)

Example:
```
>> vec = [5 8 6 4 1 3]

>> great = 19
>> out = true
```

Write your SCRIPT here:

```
s = sum(vec(1:end ./2))
p = prod(vec(end./2 +1 : end))
great = max([s p])
out = s >= p
```

Problem 4. Write the following function. Do NOT hardcode! (30 points)

Function Name: `matrixManipulator`

Inputs (2): (double) MxN array
(double) 1xN vector

Outputs (2): (double) a manipulated array
(double) a calculated value

Function Description: You are to manipulate a given MxN array and a 1xN vector.

Write a function that completes the following steps in order:

- Sort the columns of the input array in ascending order according to the values in the first row.
- Replace all negative values in the array with the equivalent positive value (i.e. -1 should become 1, -7 should become 7, etc.)
- Delete all rows that have a maximum value less than 4.
- Replace the last row in the array with the input vector.

  Note: the vector will always have the same number of columns as the array.
- Find the sum of the right half of the array and make the result the second output of your function.

  Note: the array is guaranteed to have an even number of columns.

Test Case:

```
arr =         [3     2     1     4;
              -5    -1     7     0;
             -11    -5     0     9;
               4     4     4     2;
               0     1     3     0]

vec = [3, 0, 3, 0]

>>  [manipulatedArr, num] = matrixManipulator(arr, vec)


manipulatedArr => [1     2     3     4;
                   7     1     5     0;
                   0     5    11     9;
                   3     0     3     0]

num=> 30
```

Name: ____

GTID (ex: 903XXXXXX): ____

Write your FUNCTION here:

```
function [out1 out2] = matrixManipulator(arr, vec)
    [~, ind] = sort(arr(1,:));
    arr = arr(:, ind);
    arr(arr < 0) = arr(arr < 0) .* -1;
    [m] = max(arr, [], 2);
    delete = m < 4;
    arr(delete, : ) = [];
    arr(end, :) = vec;
    right = arr(: , end./2 +1 : end);
    out2 = sum(sum(right));
    out1 = arr;
end
```

EXTRA SPACE: if you use this space, be sure to label which question you are answering. In the original question space, CLEARLY leave a note that you have continued the answer here.