```
% 1. Part 1 - Solutions of systems of linear equations
   % By considering random matrices of appropriate sizes, find "emperical
 evidence" that
   % substantiate the following statements. (In this part, m and n are both
 integers and both are greater than 6, so 7 or larger).
   % Give examples of exceptions for each case (but for this part, you can
 use integers m and n that need only be larger than 2, so 3 or larger).

   % (a) A system of n linear equations in n unknowns typically has a unique
 solution.
   % (b) A system of m linear equations in n unknowns, where m > n, typically
 has no solution.
   % (c) A system of m linear equations in n unknowns, where m < n, typically
 has many solutions.

% (a) Case 1: n*n typically has a unique solution. Generate 100 random
 matrices
           % with integers of size n x n where n is between 8 and 20.
   sprintf("Part 1A: Checking for consistency in a n*n matrix by comparing
 the " + ...
       "rank of the coefficient matrix to the rank of the augmented matrix, "
 + ...
       "if they are equal, a unique solution exists. Alternatively, check"
 + ...
       " if the RREF is equal to an n sized identity matrix.")

   count = 0;
   for i = 1:100
       % Make a selection of n, where 8 <= n <= 20
       n = randi([8, 20]);
       % Generate a matrix of size n * n with real numbers
       A = rand(n);
       % Generate a vector of size n * 1 to form our augmented matrix
       b = rand(n, 1);
       % Generate our augmented matrix by concatenating matrix A and vector b
       Ab = cat(2, A, b);

       % Verification for testing purposes only.......
       Ab_rref = rref(Ab);

       % If rank(Coeffient matrix == augmented matrix), increment count
       if rank(A) == rank(Ab)
           count = count + 1;
       end
   end
   sprintf('Out of 100 randomly generated systems of linear equations of size
 m=n, %d had a unique solution', count)

   % Print out MY generated exception to the rule
   sprintf('My exception of a m=n system without a unique solution')
   A = [1 2 3; 1 2 3; 1 2 3];
   b = [1; 2; 2];
```

```
    Ab = cat(2, A, b)
    rref(Ab)
    sprintf('End part 1A')

% (b) Case 2: m > n typically has no solution
    % Generate 100 random matrices with integers of size m x n where m is
 between 8 and 20 and n is between 8 and 20
    sprintf(['Part 1B: Checking for inconsistency in a m > n matrix by
 evaluating' ...
        'the rank of the coefficient matrix and the augmented matrix. If
 rank(coefficient) ' ...
        'is less than rank(augmented), then the system is inconsistent'])

    count2 = 0;
    for i = 1:100
        % Make a selection for m, where 15 <= m <= 20
        m = randi([15, 20]);
        % Make a selection for n, where m > n and greater than 7 (aka 7-15)
        n = randi([7, m-1]);

        % Generate a matrix of size mxn with real numbers
        B = rand(m, n);
        % Generate a vector of size m * 1 to form our augmented matrix
        c = rand(m, 1);
        % Generate our augmented matrix by concatenating matrix B and
        % vector b
        Bc = cat(2, B, c);

         % Verification for testing purposes only...
        Bc_rref = rref(Bc);

        % If rank(coefficient) < rank(augmented), increment count2
        if rank(B) < rank(Bc)
            count2 = count2 + 1;
        end
    end
    sprintf('Out of 100 randomly generated systems of linear equations of size
 m > n, %d did not have a solution', count2)

    % This is an example of a system of linear equations that does not follow
 the rule that m > n typically has no solution
    sprintf('My exception of a m > n with many solutions')
    B2 = [1 2 3; 2 4 6; 3 3 4; 10 10 12];
    c2 = [1; 2; 3; 4];
    Bc2 = cat(2, B2, c2)
    Bc_rref2 = rref(Bc2)
    sprintf('End part 1B')

% (c) Case 3: m < n typically has many solutions
    sprintf(['Part 1C: A matrix of m < n typically has many solutions. We
 will' ...
        'be first checking for consistency. If the system is consistent...
 ' ...
```

```
        'we will check that there exists a column of 0s. In other words,
that' ...
        'we have a free variable. We do this by checking the condition:' ...
        'If n - rank > 0... then there are many solutions. '])

    count3 = 0;
    for i = 1:100
        % Generate an integer for n from [12 to 20]
        n = randi([12, 20]);
        % Generate an integer from [7 to n-1]
        m = randi([7, n-1]);

        % Generate a matrix of size m * n with real numbers
        C = rand(m, n);

        % Generate a vector of size m * 1 to form our augmented matrix
        D = rand(m, 1);

        % Concentrate matrix C and vector D to create an augmented matrix
        CD = cat(2, C, D);

        % If the rank of the coefficient matrix is LESS than the rank of
        % the augmented matrix, there are no solutions
        if rank(C) < rank(CD)
            break
        end

        % For my checks only...
        Cd_rref = rref(CD);

        % If # of variables (columns) - rank is > 0, many solutions exist
        % Increment count3
        if n - rank(Cd_rref) > 0
            count3 = count3 + 1;
        end
    end
    sprintf('Out of 100 randomly generated systems of linear equations of size
m < n, %d had many solutions', count3)

    sprintf('Example of a system of linear equations (augmented matrix) that
size m < n that is inconsistent and does not have many solutions')
    m3 = 3;
    n3 = 4;
    C3 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
    D3 = [1; 2; 4];
    CD3 = cat(2, C3, D3)
    Cd_rref3 = rref(CD3)
    sprintf('End Part 1C')


% BEGIN PART2
% Consider the set of vectors in R5: B1 = {w1, w2, w3, w4, w5, w6} where
w1 = [-1; 1; 2; 4; 1];
w2 = [-1; 1; 2; 1; 1];
```

```
w3 = [3; 1; -1; 2; 0];
w4 = [2; 1; 0; 3; -1];
w5 = [5; 4; 1; 11; 3];
w6 = [1; 0; -1; 2; 1];

sprintf('Part 2A: Show B1 is a linearly dependent set.')
% Create a system of equations using the vectors in B1
B1 = [w1(:), w2(:), w3(:), w4(:), w5(:), w6(:)];
% Initialize a zero vector to form augmented matrices
zeroVectorR5 = [0; 0; 0; 0; 0];

% Create an augmented matrix wih the zero vector and row reduce it to RREF
B1Aug = cat(2, B1, zeroVectorR5);
B1RREF = rref(B1Aug)

% The system of equations is not linearly independent
sprintf('The system is consistent with a non-zero solution, x5 and x6 are
 free');

% Find a maximal linearly independent set B#1 of vectors from B1
sprintf(['The maximal linearly independent set B''1 of vectors is {w1, w2, w3,
 w4}' ...
    'as seen in the RREF x5 and x6 do not have leading 1s'])

% For testing purposes only... making sure that there is a unique solution
B1_maximal = [w1(:), w2(:), w3(:), w4(:)];
B1_maximal_rref = rref(B1_maximal);

sprintf('Show that vectors from B1 that are NOT in B#1 set are contained in
 the span of B#1 (and hence, that span B1 = span B#1).')
% I am using consistency in w5 = {w1, w2, w3, w4} and w6 = {w1, w2, w3, w4}
B1_w5_Aug = cat(2, [w1(:), w2(:), w3(:), w4(:)], w5);
B1_w5_Rref = rref(B1_w5_Aug);
B1_w6_Aug = cat(2, [w1(:), w2(:), w3(:), w4(:)], w6);
B1_w6_Aug_Rref = rref(B1_w6_Aug);
% Both are clearly consistent

% What is the dimension of span B1?
sprintf(['The dimension of the span is equal to how many independent vectors
 there are in Span B''1. In this case' ...
    'there are obviously 4 independent vectors. Thus the dimension is 4.'])

% Consider the set: B2 = {z1, z2, z3, z4, z5} where
z1 = [5; 2; 1; 7; 1];
z2 = [2; -1; 0; 0; 1];
z3 = [1; 2; 1; 1; 0];
z4 = [2; -4; -2; 4; 1];
z5 = [0; 1; 2; 3; -1];

% Create a system of equations using the vectors in B2
B2 = [z1(:), z2(:), z3(:), z4(:), z5(:)];

% Create an augmented matrix wih the zero vector and row reduce to RREF
B2Aug = cat(2, B2, zeroVectorR5);
```

```
B2RREF = rref(B2Aug)
sprintf('As seen in B2RREF, the system is consistent with a non-zero
 solution');
sprintf('There is no leading variables for x4, it is a free variable');

% Find a maximal linearly independent set B#2 of vectors from B2
sprintf('The maximal linearly independent set B''1 of vectors is {z1, z2, z3,
 z5}')

% For testing purposes only... making sure that there is a unique solution
B2_maximal = [z1(:), z2(:), z3(:), z5(:)];
B2_maximal_rref = rref(B2_maximal);

% Find all the vectors in the intersection span(B1) INTERSECTION span(B2).
% I am using the maximal linearly independent sets of B1 and B2
% Set the linearly combination of both spans equal to each other.
B1B2_Maximal_System = [w1(:), w2(:), w3(:) w4(:), -z1(:), -z2(:), -z3(:), -
z5(:)]
B1B2_Aug = cat(2, B1B2_Maximal_System, zeroVectorR5)
B1B2_Rref = rref(B1B2_Aug)
B1B2_Rref = rats(B1B2_Rref)

intersection1 = rats([-7/9 ; -5/9; -4/3; -1; -4/3]);
intersection2 = rats([-4/9; 29/18; 11/12; 3/4; 5/12;]);
intersection3 = rats([-2/9; 5/9; -7/6; 3/2; -1/6]);
```

```
ans =

    "Part 1A: Checking for consistency in a n*n matrix by comparing the rank
 of the coefficient matrix to the rank of the augmented matrix, if they are
 equal, a unique solution exists. Alternatively, check if the RREF is equal to
 an n sized identity matrix."


ans =

    'Out of 100 randomly generated systems of linear equations of size m=n,
 100 had a unique solution'


ans =

    'My exception of a m=n system without a unique solution'


Ab =

     1     2     3     1
     1     2     3     2
     1     2     3     2


ans =
```

```
     1     2     3     0
     0     0     0     1
     0     0     0     0


ans =

    'End part 1A'


ans =

    'Part 1B: Checking for inconsistency in a m > n matrix by evaluatingthe
 rank of the coefficient matrix and the augmented matrix. If rank(coefficient)
 is less than rank(augmented), then the system is inconsistent'


ans =

    'Out of 100 randomly generated systems of linear equations of size m > n,
 100 did not have a solution'


ans =

    'My exception of a m > n with many solutions'


Bc2 =

     1     2     3     1
     2     4     6     2
     3     3     4     3
    10    10    12     4


Bc_rref2 =

    1.0000         0         0    2.5000
         0    1.0000         0   -7.5000
         0         0    1.0000    4.5000
         0         0         0         0


ans =

    'End part 1B'


ans =

    'Part 1C: A matrix of m < n typically has many solutions. We willbe first
 checking for consistency. If the system is consistent... we will check that
```

```
 there exists a column of 0s. In other words, thatwe have a free variable.
 We do this by checking the condition:If n - rank > 0... then there are many
 solutions. '


ans =

    'Out of 100 randomly generated systems of linear equations of size m < n,
 100 had many solutions'


ans =

    'Example of a system of linear equations (augmented matrix) that size m <
 n that is inconsistent and does not have many solutions'


CD3 =

     1     2     3     4     1
     5     6     7     8     2
     9    10    11    12     4


Cd_rref3 =

     1     0    -1    -2     0
     0     1     2     3     0
     0     0     0     0     1


ans =

    'End Part 1C'


ans =

    'Part 2A: Show B1 is a linearly dependent set.'


B1RREF =

     1     0     0     0     2     1     0
     0     1     0     0     0    -1     0
     0     0     1     0     3     1     0
     0     0     0     1    -1    -1     0
     0     0     0     0     0     0     0


ans =

    'The maximal linearly independent set B'1 of vectors is {w1, w2, w3, w4}as
 seen in the RREF x5 and x6 do not have leading 1s'
```

```
ans =

    'Show that vectors from B1 that are NOT in B#1 set are contained in the
 span of B#1 (and hence, that span B1 = span B#1).'


ans =

    'The dimension of the span is equal to how many independent vectors there
 are in Span B'1. In this casethere are obviously 4 independent vectors. Thus
 the dimension is 4.'


B2RREF =

     1     0     0     1     0     0
     0     1     0     0     0     0
     0     0     1    -3     0     0
     0     0     0     0     1     0
     0     0     0     0     0     0


ans =

    'The maximal linearly independent set B'1 of vectors is {z1, z2, z3, z5}'


B1B2_Maximal_System =

    -1    -1     3     2    -5    -2    -1     0
     1     1     1     1    -2     1    -2    -1
     2     2    -1     0    -1     0    -1    -2
     4     1     2     3    -7     0    -1    -3
     1     1     0    -1    -1    -1     0     1


B1B2_Aug =

    -1    -1     3     2    -5    -2    -1     0     0
     1     1     1     1    -2     1    -2    -1     0
     2     2    -1     0    -1     0    -1    -2     0
     4     1     2     3    -7     0    -1    -3     0
     1     1     0    -1    -1    -1     0     1     0


B1B2_Rref =

  Columns 1 through 7

    1.0000         0         0         0         0    0.7778    0.4444
         0    1.0000         0         0         0    0.5556   -1.6111
         0         0    1.0000         0         0    1.3333   -0.9167
```

```
         0         0         0    1.0000         0    1.0000   -0.7500
         0         0         0         0    1.0000    1.3333   -0.4167

  Columns 8 through 9

    0.2222         0
   -0.5556         0
    1.1667         0
   -1.5000         0
    0.1667         0


B1B2_Rref =

  5×126 char array

    '       1             0             0             0             0
   7/9          4/9           2/9           0      '
    '       0             1             0             0             0
   5/9        -29/18         -5/9           0      '
    '       0             0             1             0             0
   4/3        -11/12          7/6           0      '
    '       0             0             0             1             0
    1          -3/4          -3/2           0      '
    '       0             0             0             0             1
   4/3         -5/12          1/6           0      '
```

Published with MATLAB® R2022a