Assignment 3

Name: Mohd Darish Khan

Roll: 2101MC29

Task1 :

Code :

```
% part 1 : for interval [1, inf)

syms f(x)
f(x) = x^2;
gradf = gradient(f);

disp("function is x^2, with interval [1, inf)")
% for point x =1
x = 1;
if x-1==0
    d = 1;
    if gradf(x)*d >= 0
        fprintf("function satisfies FONC at x = %d\n", x);
    end
elseif x-1>0
    d = 1;
    flag = 0;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    d=-1;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    if flag == 2
        fprintf("function satisfies FONC at x = %d\n", x);
    else
        fprintf("function doesn't satisfies FONC at x = %d\n", x);
    end
else
    fprintf("Point %d is not feasible\n", x);
end




x=2;
if x-1==0
    d = 1;
    if gradf(x)*d >= 0
```

```
        fprintf("function satisfies FONC at x = %d\n", x);
    end
elseif x-1>0
    d = 1;
    flag = 0;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    d=-1;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    if flag == 2
        fprintf("function satisfies FONC at x = %d\n", x);
    else
        fprintf("function doesn't satisfies FONC at x = %d\n", x);
    end
else
    fprintf("Point %d is not feasible\n", x);
end


% part 2 : for interval [-1, inf)

syms x
f(x) = x^2;
gradf = gradient(f);
fprintf("\n\n");
disp("function is x^2, with interval [-1, inf)")
% for point x =1
x = 1;
if x+1==0
    d = 1;
    if gradf(x)*d >= 0
        fprintf("function satisfies FONC at x = %d\n", x);
    end
elseif x+1>0
    d = 1;
    flag = 0;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    d=-1;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    if flag == 2
        fprintf("function satisfies FONC at x = %d\n", x);
    else
        fprintf("function doesn't satisfies FONC at x = %d\n", x);
    end
```

```
else
    fprintf("Point %d is not feasible\n", x);
end




x=2;
if x-1==0
    d = 1;
    if gradf(x)*d >= 0
        fprintf("function satisfies FONC at x = %d\n", x);
    end
elseif x-1>0
    d = 1;
    flag = 0;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    d=-1;
    if gradf(x)*d == 0
        flag = flag + 1;
    end
    if flag == 2
        fprintf("function satisfies FONC at x = %d\n", x);
    else
        fprintf("function doesn't satisfies FONC at x = %d\n", x);
    end
else
    fprintf("Point %d is not feasible\n", x);
end
```

Output :

```
function is x^2, with interval [1, inf)
function satisfies FONC at x = 1
function doesn't satisfies FONC at x = 2


function is x^2, with interval [-1, inf)
function doesn't satisfies FONC at x = 1
function doesn't satisfies FONC at x = 2
```

Task 2:

Code :

```
% Define matrix A
A = [1 4 5; 4 3 2; 1 0 1];
rank_A = rank(A);

% Check if A has full rank
is_full_rank_A = (rank_A == size(A, 2));

% Display results
fprintf('Matrix A:\n');
disp(A);
fprintf('Rank of A: %d\n', rank_A);
if is_full_rank_A
    fprintf('Matrix A has full rank.\n');
else
    fprintf('Matrix A does not have full rank.\n');
end

% Find the linearly independent columns
rref_A = rref(A);
linearly_independent_cols = find(any(rref_A, 1));
disp("Linearly independent columns of B are")
for i = linearly_independent_cols
    disp(A(:, i))
end

% -----------------------------------------------------------


% Define matrix B
B = [1 8 5; 5 0 1; 1 0 90];
rank_B = rank(B);

% Check if B has full rank
is_full_rank_B = (rank_B == size(B, 2));

fprintf('Matrix B:\n');
disp(B);
fprintf('Rank of B: %d\n', rank_B);
if is_full_rank_B
    fprintf('Matrix B has full rank.\n');
else
    fprintf('Matrix B does not have full rank.\n');
end

% Find the linearly independent columns
rref_B = rref(B);
```

```
linearly_independent_cols = find(any(rref_B, 1));
disp("Linearly independent columns of B are")
for i = linearly_independent_cols
    disp(B(:, i))
end
```

Output :

```
Matrix A:
     1     4     5
     4     3     2
     1     0     1

Rank of A: 3
Matrix A has full rank.
Linearly independent columns of B are
     1
     4
     1

     4
     3
     0

     5
     2
     1

Matrix B:
     1     8     5
     5     0     1
     1     0    90

Rank of B: 3
Matrix B has full rank.
Linearly independent columns of B are
     1
     5
     1

     8
     0
     0

     5
     1
    90
```

Task 3:

Code :

```
disp("Part 1 : ")
A_a = [1 1; 2 2];
b_a = [2; 4];

Ab_a = [A_a b_a];

rref_A_a = rref(Ab_a);

A_rref_a = rref_A_a(:, 1:end-1);
b_rref_a = rref_A_a(:, end);

basis_a = null(A_a, 'r');

fprintf('Basis for the solution space of system a:\n');
disp(basis_a);

fprintf("\n");

% ----------------------------------------------------------

disp("Part 2 : ")
A_b = [1 1 -1; 1 -1 -1; 2 3 4];
b_b = [3; 4; 0];

Ab_b = [A_b b_b];

rref_A_b = rref(Ab_b);

A_rref_b = rref_A_b(:, 1:end-1);
b_rref_b = rref_A_b(:, end);

basis_b = null(A_b, 'r');

fprintf("Basis for the solution space of system b doesn't exists
\n");
fprintf("solution is unique\n")
disp(basis_b);

fprintf('\n');

% -----------------------------------------------------------

disp("Part 3 : ")
A_c = [1 1 1 1; 1 2 2 1; 2 3 3 2];
b_c = [4; 8; 10];
```

```
Ab_c = [A_c b_c];

rref_A_c = rref(Ab_c);

A_rref_c = rref_A_c(:, 1:end-1);
b_rref_c = rref_A_c(:, end);

basis_c = null(A_c, 'r');

fprintf('Basis for the solution space of system c:\n');
disp(basis_c);
```

Output :

```
Part 1 :
Basis for the solution space of system a:
    -1
     1


Part 2 :
Basis for the solution space of system b doesn't exists
solution is unique

Part 3 :
Basis for the solution space of system c:
     0    -1
    -1     0
     1     0
     0     1
```