Ramaneek Gill
1000005754

CSC336 Assignment 3.

QUESTION 1:

a)

code:

```
////////////////////////////////////////////////////////////////////////////////
format long;
warning('off', 'all');

b = zeros(2,1);
b(2,1) = 1;

L = zeros(2,2);
L(1,1) = 1;
L(2,2) = 1;

U = zeros(2,2);
U(1,2) = 1;

k = 1;
while (k <= 10)
	gamma = 10^(-2*k);
	b(1,1) = 2-gamma;
	L(2,1) = 1/gamma;
	U(1,1) = gamma;
	U(2,2) = 1 - (1/gamma);

	y = L\b;
	x = U\y;

	disp(['k is ', num2str(k), ' x is ']);
	disp(x);

	k = k + 1;
end
////////////////////////////////////////////////////////////////////////////////
```

OUTPUT:

```
k is 1 x is
  -1.00000000000000
   2.00000000000000

k is 2 x is
```

```
  -0.99999999999989
   2.00000000000000

k is 3 x is
  -0.99999999991773
   2.00000000000000

k is 4 x is
  -0.99999999392253
   2.00000000000000

k is 5 x is
  -1.00000008274037
   2.00000000000000

k is 6 x is
  -1.00008890058234
   2.00000000000000

k is 7 x is
  -0.99920072216264
   2.00000000000000

k is 8 x is
     0
     2

k is 9 x is
     0
     2

k is 10 x is
     0
     2
```

As the value of gamma reduces it seems as if we are running into greater rounding errors. This becomes highly evident when k's value is greater than or equal to 8 since the answer is completely wrong which results in a very high absolute and relative error.

b)

code:

```
////////////////////////////////////////////////////////////////////////////////
format long;
warning('off', 'all');

b = zeros(2,1);
```

```
b(2,1) = 1;

p = zeros(2,2);
p(1,2) = 1;
p(2,1) = 1;

L = zeros(2,2);
L(1,1) = 1;
L(2,2) = 1;

U = zeros(2,2);
U(1,1) = 1;
U(1,2) = 1;

k = 1;
while (k <= 10)
        gamma = 10^(-2*k);
        b(1,1) = 2-gamma;
        L(2,1) = gamma;
        U(2,2) = 1-gamma;

        b_hat = p*b;

        y = L\b_hat;
        x = U\y;

        disp(['k is ', num2str(k), ' x is ']);
        disp(x);

        k = k + 1;
end
////////////////////////////////////////////////////////////////////////////////

OUTPUT:
k is 1 x is
    -1
     2

k is 2 x is
    -1
     2

k is 3 x is
  -1.00000000000000
   2.00000000000000

k is 4 x is
  -1.00000000000000
```

```
   2.00000000000000

k is 5 x is
    -1
     2

k is 6 x is
  -1.00000000000000
   2.00000000000000

k is 7 x is
    -1
     2

k is 8 x is
  -1.00000000000000
   2.00000000000000

k is 9 x is
    -1
     2

k is 10 x is
    -1
     2
```

Unlike the algorithm in part a if we include a permutation matrix in our calculations then we can avoid rounding errors and catastrophic cancellation altogether and have a consistent solution throughout each iteration.

c)

code:

```
////////////////////////////////////////////////////////////////////////////////
format long;
warning('off', 'all');

b = zeros(2,1);
b(2,1) = 1;

L = zeros(2,2);
L(1,1) = 1;
L(2,2) = 1;

U = zeros(2,2);
U(1,2) = 1;

k = 1;
```

```
while (k <= 10)
            gamma = 10^(-2*k);
            b(1,1) = 2-gamma;
            L(2,1) = 1/gamma;
            U(1,1) = gamma;
            U(2,2) = 1 - (1/gamma);

            y = L\b;
            x_hat = U\y;

            A = L*U;
            r = b-A*x_hat;

            z = L\r;
            e = U\z;

            x_tilda = x_hat + e;

            disp(['k is ', num2str(k), ' x_tilda is ']);
            disp(x_tilda);

            k = k + 1;
end
////////////////////////////////////////////////////////////////////////////////
```

```
OUTPUT:
k is 1 x_tilda is
   -1
    2

k is 2 x_tilda is
   -1
    2

k is 3 x_tilda is
   -1
    2

k is 4 x_tilda is
   -1
    2

k is 5 x_tilda is
   -1
    2

k is 6 x_tilda is
  -1.00000000000000
```

```
   2.00000000000000

k is 7 x_tilda is
    -1
     2

k is 8 x_tilda is
     1
     2

k is 9 x_tilda is
     1
     2

k is 10 x_tilda is
     1
     2
```

This algorithm's output is almostthe exact same as A, a key distinction here is that through our first 7 operations part c is more accurate than part a. Afterwards they both produce a huge error relative to the input because of rounding errors/catastrophic cancellation.

QUESTION 2:

code:

```
////////////////////////////////////////////////////////////////////////////////
n = 60;
A = ones(n,n);
A = A - triu(A);
A = eye(n) - A;
A = A + [ones(n-1, 1); 0] * [zeros(1,n-1),1];
Q =diag(ones(n-1,1),1);
Q(n-1) = 1;

[L1, U1, P1] = lu(A);
U1(n,n)  %for verifying it is == 2^(n-1)
[L2, U2] = lu(A*Q);
max(max(abs(U2)))  %for verifying == 2

x = ones(n,1);
b = A*x;
y = L1\b;
x1 = U1\y;
norm(x-x1, inf)

y2 = L2\b;
z = U2\y2;
```

```
x2 = Q * z;
norm(x-x2, inf)
////////////////////////////////////////////////////////////////////////////////
```

OUTPUT:

```
ans =

   5.764607523034235e+17


ans =

   2


ans =

   1


ans =

  NaN
```

QUESTION 4:

code:

```
////////////////////////////////////////////////////////////////////////////////

function y = perm_a(p, x)
        p_length = size(p);
        p_length = max(p_length(1), p_length(2));
        i = 1;
        y=x;

        while (i <= p_length)
                y([i p(i)],:) = y([p(i) i],:);
                i = i + 1;
        end

end

////////////////////////////////////////////////////////////////////////////////

function q = perm_b(p)
```

```
        p_length = size(p);
        p_length = max(p_length(1), p_length(2));

        x = eye(p_length + 1);
        q = zeros(1, p_length + 1);
        i = 1;

        while (i <= p_length)
                x([i p(i)],:) = x([p(i) i],:);
                i = i + 1;
        end

        i = 1;
        x_size = size(x);

        while(i <= x_size(1))
                j = 1;
                while(j <= x_size(2))
                        if(x(i,j) == 1)
                                q(i) = j;
                        end
                        j = j + 1;
                end
                i = i + 1;
        end
end

////////////////////////////////////////////////////////////////////////////////

function y = perm_c(q,x)
        q_size = size(q);
        q_size = max(q_size(1), q_size(2));
        p = zeros(size(q));
        i = 1;
        while(i <= q_size)
                j = 1;
                while(j <= q_size)
                        if(j == q(i))
                                p(i,j) = 1;
                        end
                        j = j + 1;
                end
                i = i + 1;
        end
        y = p*x;

end
```

//

OUTPUT:

```
>> p
p =
            3     5     9     4    10     8     7     9    10

>> x
x =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10

>> y1 = perm_a(p,x)
y1 =
     3
     5
     9
     4
    10
     8
     7
     1
     2
     6

>> q = perm_b(p)
q =
     3     5     9     4    10     8     7     1     2     6

>> y2 = perm_c(q, x)
y2 =
     3
     5
     9
     4
    10
     8
     7
     1
```

2
6