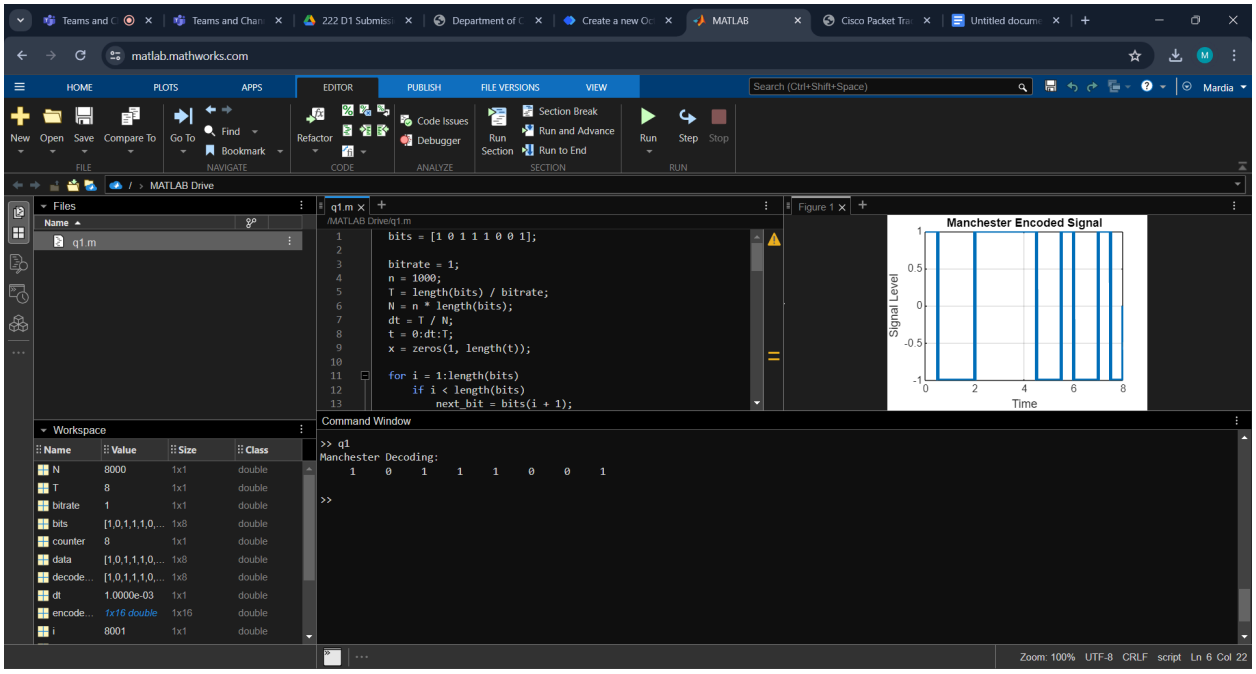
1.

```
bits = [1 0 1 1 1 0 0 1];

bitrate = 1;
n = 1000;
T = length(bits) / bitrate;
N = n * length(bits);
dt = T / N;
t = 0:dt:T;
x = zeros(1, length(t));

for i = 1:length(bits)
    if i < length(bits)
        next_bit = bits(i + 1);
    else
        next_bit = 0;
    end

    if bits(i) == 1
        if next_bit == 0
            x((i-1)*n+1:(i-1)*n+n/2) = 1;
            x((i-1)*n+n/2:i*n) = -1;
        else
            x((i-1)*n+1:(i-1)*n+n/2) = 1;
            x((i-1)*n+n/2:i*n) = 1;
        end
    else
        if next_bit == 0
            x((i-1)*n+1:(i-1)*n+n/2) = -1;
            x((i-1)*n+n/2:i*n) = 1;
        else
            x((i-1)*n+1:(i-1)*n+n/2) = -1;
            x((i-1)*n+n/2:i*n) = -1;
        end
    end
end

figure;
plot(t, x, 'Linewidth', 3);
title('Manchester Encoded Signal');
xlabel('Time');
ylabel('Signal Level');
grid on;
```

```
counter = 0;
result = [];

for i = 1:length(t)
    if t(i) > counter
        counter = counter + 1;
        if x(i) > 0
            result = [result 1];
        else
            result = [result 0];
        end
    end
end

disp('Manchester Decoding:');
disp(result);


disp(decoded_data);
```




3.

```
#include <stdio.h>
#include <math.h>

int calculate_parity_bits(int data_bits) {
```

```
    int r = 0;
    while ((1 << r) < (data_bits + r + 1)) {
        r++;
    }
    return r;
}

void generate_hamming_code(int data[], int data_bits, int total_bits) {
    int parity_bits = calculate_parity_bits(data_bits);
    int encoded_data[total_bits];
    int j = 0, k = 0;

    for (int i = 0; i < total_bits; i++) {
        encoded_data[i] = 0;
    }

    for (int i = 1; i <= total_bits; i++) {
        if (i == (1 << j)) {
            j++;
        } else {
            encoded_data[i - 1] = data[k++];
        }
    }

    for (int i = 0; i < parity_bits; i++) {
        int position = (1 << i);
        int count = 0;

        for (int j = position; j <= total_bits; j += (position << 1)) {
            for (int k = 0; k < position && (j + k) <= total_bits; k++) {
                if (encoded_data[j + k - 1] == 1) {
                    count++;
                }
            }
        }
        encoded_data[position - 1] = count % 2;
    }

    printf("Encoded data: ");
    for (int i = 0; i < total_bits; i++) {
        printf("%d ", encoded_data[i]);
    }
    printf("\n");
}
```

```
int main() {
    int data_bits;

    printf("Enter the number of data bits: ");
    scanf("%d", &data_bits);

    int data[data_bits];

    printf("Enter the data bits: ");
    for (int i = 0; i < data_bits; i++) {
        scanf("%d", &data[i]);
    }

    int parity_bits = calculate_parity_bits(data_bits);
    int total_bits = data_bits + parity_bits;

    generate_hamming_code(data, data_bits, total_bits);

    return 0;
}
```