Part D

We first process each frame to binary

```
In [ ]: frame1 = bin(0x127D555503876B92FF)
        frame2 = bin(0x127EAAAA046798B912FF)
        frame3 = bin(0x27EAAAA059867F922FF)

        frames = [frame1, frame2, frame3]
        n_s = [(len(frame)//4 + 1)*4 for frame in frames ]
        processed_frames = [frames[i][2:].rjust(n_s[i], '0') for i in range(len(f
```

We define the decode() function to iterate through the frame and pick out the id and data and ensure all other forms are as expected

```
In [ ]: def decode(frame):
            pointer = 0
            id = ''
            if frame[pointer] != '0':
                print('Start of frame bit error')
            pointer +=1

            for i in range(11):
                id += frame[pointer]
                pointer +=1

            if frame[pointer] != '1':
                print('SRR bit error')
            pointer +=1

            if frame[pointer] != '1':
                print('IDE bit error')
            pointer +=1

            for i in range(18):
                id += frame[pointer]
                pointer +=1


            if frame[pointer] != '0':
                print('It is not a data frame')
            pointer +=1

            pointer += 2 # reserved bits we dont care about

            num_bytes = int(frame[pointer: pointer+4], 2)
            pointer +=4

            data = frame[pointer: pointer+(num_bytes*8)]
            pointer += (num_bytes*8)

            crc = frame[pointer: pointer+15]
            pointer +=15

            if frame[pointer] != '1':
                print('crc delimiter error')
```

```
    pointer +=1

    pointer +=1 #ack slot, we dont care about this for the scope of the q

    if frame[pointer] != '1':
        print('ack delimiter error')
    pointer +=1

    if frame[pointer: pointer +7] != '1111111':
        print('end of frame error')
    pointer +=7

    return id, data
```

We print the id and data for each frame.

In [ ]:
```
for i in range(3):
    id, data = decode(processed_frames[i])
    print(f'Frame {i+1}:')
    print(f'ID: {id}')
    print(f'Data: {data}')
```

```
Frame 1:
ID: 00100100111010101010101010101
Data: 11000011
Frame 2:
ID: 00100100111101010101010101010
Data: 0011001111001100
Frame 3:
ID: 00000100111101010101010101010
Data: 1100110000110011
```

Looking at this we can say the priority order of frame 3 > frame 1 > frame 2.