ECGR 4101/5101 LECTURE 15

Q_T rx_q, tx_q;

(1)

0	1	2	3	4	5	6	7
x48	x45	x52	x20	x20	x20	x20	x20

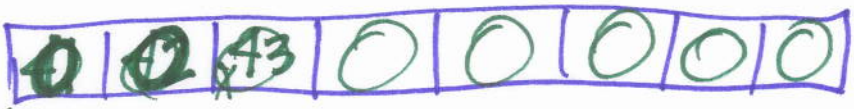
rx_q → Data[i];
rx_q -> Head;
rx_q -> Tail;
rx_q -> Size;
t = x48;

Receive x48 ✓
Receive x45 ✓
Consume one ✓
Receive x52 ✓
~~Consume one~~
Receive x20 ✓
x20 ✓
x20 ✓
x20 ✓
x20 ✓
x20 ✓
x20
x20

ECOR 4101/5101 LECTURE 15

Receive x41 ✓
Consume Return x41
Receive x42 ✓
Consume Return x42
Consume Return 0
Receive x43

Now, show me the following... (2)



rx_q -> Data[i]
rx_q -> Head 0x2
rx_q -> Tail 0x3
rx_q -> Size 0x0x0x0 1

ECGR 4101/5101 LECTURE 15

3

```
// Quiz 11 code

#define Q_SIZE (8)
typedef struct {
  unsigned char Data[Q_SIZE];
  unsigned int Head; // points to oldest data element
  unsigned int Tail; // points to next free space
  unsigned int Size; // quantity of elements in queue
} Q_T;
Q_T tx_q, rx_q;

void Q_Init(Q_T * q) {
  unsigned int i;
  for (i=0; i<Q_SIZE; i++)
    q->Data[i] = 0;  // to simplify our lives when debugging
  q->Head = 0;
  q->Tail = 0;
  q->Size = 0;
}
int Q_Empty(Q_T * q) {
  return q->Size == 0;
}
int Q_Full(Q_T * q) {
  return q->Size == Q_SIZE;
}

// Q_Enqueue - Called by a UART ISR - put a char on the queue
int Q_Enqueue(Q_T * q, unsigned char d) {
  if (!Q_Full(q)) { // What if queue is full?
    q->Data[q->Tail++] = d;
    q->Tail %= Q_SIZE;
    q->Size++;
    return 1; // success
  } else
    return 0; // failure
}
// Q_Dequeue-called by a consumer function-take a char from queue
unsigned char Q_Dequeue(Q_T * q) {
  unsigned char t=0;
  if (!Q_Empty(q)) {    // Must check to see if queue is empty 1st
    t = q->Data[q->Head];
    q->Data[q->Head++] = 0;   // to simplify debugging, clear
    q->Head %= Q_SIZE;
    q->Size--;
  }
  return t;
}
```