M. G. Gouda	CS 356
Summer 2006	Midtem 2

1. (10 points)
Modify the cumulative acknowledgment protocol in Section 9.3 as follows. Process p can send two types of data messages: regular and short. Each data message, whether regular or short, is of the form data(i), where i is the unique message number of the message. Process p does not send more than w data messages, whether regular or short, without receiving an acknowledgement message for any of them. Moreover, p does not send more than one short data message without receiving an acknowledgement message for all previous data messages. Specify process p in this protocol. Please add the two comments {send short message} and {send long message}, where appropriate, to your process p.

2. (10 points)
Design a protocol that performs forward recovery from n-bounded reorder. The protocol consists of two processes p and q. Process p sends an (infinite) stream of data(s,t) messages, where s is the sequence number of the message in the range 0..2n-1, and t is an arbitrary integer that constitutes the message text. Process q has a variable "exp" whose value, in the range 0..2n-1, is the sequence number of the next expected message from process p. When process q receives a data(s,t) message and detects that exp=s, q increments exp by 1 modulo 2n, and stores t in the next available position in an infinite array "txt" declared as follows:

```
var   txt: array [integer] of integer,
      x  :  integer                        {index of txt, init. 0}
```

When process q receives a data(s,t) message and detects that exp != s, then q leaves exp unchanged, and stores t in an circular buffer defined by the following two arrays:

```
var   rcvd : array [0..2n-1] of boolean,   {init. false}
      rcvtxt : array [0..2n-1] of integer
```

Process q is specified as follows:

```
process q
const n
var txt: array [integer] of integer,
    x  :  integer                          {index of txt, init. 0}
    rcvd : array [0..2n-1] of boolean,     {init. false}
    rcvtxt : array [0..2n-1] of integer
    s : 0..2n-1,
    t: integer
begin
rcv data(s,t) from p  -> S
end
```

Specify statement S in process q.