```
#define MAX_SEQ 16
typedef enum {frame_arrival, cksum_err, timeout} event_type;
include “protocol.h”
void sender_par_cum2 (void)
{seq_nr  nfs_odd, nfs_even;
 frame s,r;
 packet buffer_odd, buffer_even;
 event_type event;
 nfs_odd = MAX_SEQ - 1;
 Get_Two_Buffers;      /*Macro Call*/
 while (true)
 { s.info = buffer_odd;
   s.seq  = nfs_odd;
   to_physical_layer (&s);
   s.info = buffer_even;
   s.seq  = nfs_even;
   to_physical_layer (&s);
    start_timer (nfs_even);
    wait_for_event (&event);
     if (event == frame_arrival)
      { from_physical_layer(&r);
        if (r.ack ==nfs_even)
        {  stop_timer(nfs_even);
            Get_Two_Buffers; }
      }
  } /* Note – cksum_err, timeout and ACK-mismatch all come here!
}
Macro  Get_Two_Buffers
{  from_network_layer(&buffer_odd);
   from_network_layer(&buffer_even);
   nfs_odd  = (nfs_odd + 2) MOD MAX_SEQ;
   nfs_even =  nfs_odd +1;   }
```

```
void receiver_par_cum2 (void)
{
 seq_nr  frame_expected, last_cum_ACK;
 frame r, s;
 event_type event;
 frame_expected = 1;
 last_cum_ACK = 0;
 while (true)
 {  wait_for_event (&event);
    if (event == frame_arrival)
    {  from_physical_layer (&r);
       if (r.seq == frame_expected)
       {  to_network_layer(&r.info);
          if ((r.seq MOD 2) == 0)  last_cum_ACK = r.seq;
          inc (frame_expected);
          frame_expected = frame_expected MOD MAX_SEQ;
       }
       if ((r.seq MOD 2) == 0)     /*ONLY send ACK on even frame received*/
       {
         s.ack = last_cum_ACK
         to_physical_layer (&s);
       }
    }  /* cksum_err comes here */
 }
}
```