22C:050 HW#7 solutions

9.1 The code in Figure 9.12 does not allocate a deblocking buffer initially, and only allocates one if it is readed. How can a user read the contents of a tape without ever allocating such a register?

One solution is to read byte by byte without buffering. But for a block device, it means very low efficiency.

Another solution is that the user provides a buffer with maximum length for one read operation, i.e., at least the length of one block. Therefore, the buffer is maintained by users.

10.1 Convert the following applications program to the use of a main polling loop, showing your answer in the style of Figure 10.4.

```
repeat
    read( ch );
    if ch = '('
        then repeat read( ch ) until ch = ')'
        else write( ch );
forever;
```

```
procedure application;
{ ch is global, but not used elsewhere }
{ state is global, not used elsewhere, and has initial value 1 }
begin
    case state of
      1: if not empty( inputqueue ) then begin { read a character }
                dequeue( inputqueue, ch );
                if ch = '('
                      then state := 2
                      else state := 3
        end;

      2: if not empty(inputqueue) then begin { read a charater}
                dequeue (inputqueue, ch );
                if ch = ')'
                      then state := 1
                      else state := 2
        end;

      3: if not full( outputqueue ) then begin { write ch }
```

```
                    enqueue( outputqueue, ch );
                    state := 1;
               end;
          end { case };
     end { application };
```

10.8 In Figure 10.16, the only code given is for output. Write the missing code to support character sequential input

```
char comread( struct filevariable * f )
{
     char c

     struct comportvariable * cp = (struct comportvariable *) f;
     int ie;

     while (!empty( cp->inq ))
     c = denqueue( cp->enq);

     disableints();
     ie = inp( cp->comie );
     ie = ie | TxIE;
     outp( cp->comie, ie );
     enableints();

     return c
}
```

10.10 Describe a heuristic appropriate for adjusting the length of output queues in the style of the Demos input queue heuristic given here.

If the user program fills an output queue it is too small and the limit should be incremented. If the interrupt service routine empties the queue it is to high and the limit should be decremented, but not below 2.