Problem 1

a) With a connection-oriented network, every router failure will involve the routing of that connection. At a minimum, this will require the router that is "upstream" from the failed router to establish a new downstream part of the path to the destination node, with all of the requisite signaling involved in setting up a path. Moreover, all of the routers on the initial path that are downstream from the failed node must take down the failed connection, with all of the requisite signaling involved to do this.

   With a connectionless datagram network, no signaling is required to either set up a new downstream path or take down the old downstream path. We have seen, however, that routing tables will need to be updated (e.g., either via a distance vector algorithm or a link state algorithm) to take the failed router into account. We have seen that with distance vector algorithms, this routing table change can sometimes be localized to the area near the failed router. Thus, a datagram network would be preferable. Interestingly, the design criteria that the initial ARPAnet be able to function under stressful conditions was one of the reasons that datagram architecture was chosen for this Internet ancestor.

b) In order for a router to maintain an available fixed amount of capacity on the path between the source and destination node for that source-destination pair, it would need to know the characteristics of the traffic from all sessions passing through that link. That is, the router must have per-session state in the router. This is possible in a connection-oriented network, but not with a connectionless network. Thus, a connection-oriented VC network would be preferable.

c) In this scenario, datagram architecture has more control traffic overhead. This is due to the various packet headers needed to route the datagrams through the network. But in VC architecture, once all circuits are set up, they will never change. Thus, the signaling overhead is negligible over the long run.

Problem 10

a)

Prefix Match	**Link Interface**
11100000 00	0
11100000 01000000	1
1110000	2
11100001 1	3
otherwise	3

b) Prefix match for first address is 5th entry: link interface 3
Prefix match for second address is 3nd entry: link interface 2
Prefix match for third address is 4th entry: link interface 3

Problem 12

Destination Address Range	**Link Interface**
11000000 through (32 addresses) 11011111	0
10000000 through(64 addresses) 10111111	1
11100000 through (32 addresses) 11111111	2
00000000 through (128 addresses) 01111111	3

Problem 13

223.1.17.0/26
223.1.17.128/25
223.1.17.192/28

Problem 16

Any IP address in range 128.119.40.128 to 128.119.40.191

Four equal size subnets: 128.119.40.64/28, 128.119.40.80/28, 128.119.40.96/28, 128.119.40.112/28

Problem 19

The maximum size of data field in each fragment = 680 (because there are 20 bytes IP header). Thus the number of required fragments $= \left\lceil \frac{2400-20}{680} \right\rceil = 4$

Each fragment will have Identification number 422. Each fragment except the last one will be of size 700 bytes (including IP header). The last datagram will be of size 360 bytes (including IP header). The offsets of the 4 fragments will be 0, 85, 170, 255. Each of the first 3 fragments will have flag=1; the last fragment will have flag=0.

Problem 30

a) Dx(w) = 2, Dx(y) = 4, Dx(u) = 7

b) First consider what happens if c(x,y) changes. If c(x,y) becomes larger or smaller (as long as c(x,y) >=1) , the least cost path from x to u will still have cost at least 7. Thus a change in c(x,y) (if c(x,y)>=1) will not cause x to inform its neighbors of any changes.
If c(x,y)= δ<1, then the least cost path now passes through y and has cost δ+6.

Now consider if c(x,w) changes. If c(x,w) = $\varepsilon \le 1$, then the least-cost path to u continues to pass through w and its cost changes to 5 + ε; x will inform its neighbors of this new cost. If c(x,w) = $\delta > 6$, then the least cost path now passes through y and has cost 11; again x will inform its neighbors of this new cost.

c) Any change in link cost c(x,y) (and as long as c(x,y) >=1) will not cause x to inform its neighbors of a new minimum-cost path to u .

Problem 34

a)

Router z	Informs w, $D_z(x)=\infty$
	Informs y, $D_z(x)=6$

Router w	Informs y, $D_w(x)=\infty$
	Informs z, $D_w(x)=5$
Router y	Informs w, $D_y(x)=4$
	Informs z, $D_y(x)=4$

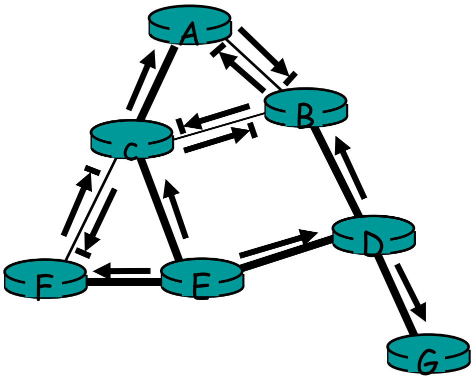
b) Yes, there will be a count-to-infinity problem. The following table shows the routing converging process. Assume that at time t0, link cost change happens. At time t1, y updates its distance vector and informs neighbors w and z. In the following table, "→" stands for "informs".

time	t0	t1	t2	t3	t4
Z	→ w, $D_z(x)=\infty$ → y, $D_z(x)=6$		No change	→ w, $D_z(x)=\infty$ → y, $D_z(x)=11$	
W	→ y, $D_w(x)=\infty$ → z, $D_w(x)=5$		→ y, $D_w(x)=\infty$ → z, $D_w(x)=10$		No change
Y	→ w, $D_y(x)=4$ → z, $D_y(x)=4$	→ w, $D_y(x)=9$ → z, $D_y(x)= \infty$		No change	→ w, $D_y(x)=14$ → z, $D_y(x)= \infty$

We see that w, y, z form a loop in their computation of the costs to router x. If we continue the iterations shown in the above table, then we will see that, at t27, z detects that its least cost to x is 50, via its direct link with x. At t29, w learns its least cost to x is 51 via z. At t30, y updates its least cost to x to be 52 (via w). Finally, at time t31, no updating, and the routing is stabilized.

time	t27	t28	t29	t30	t31
Z	→ w, $D_z(x)=50$ → y, $D_z(x)=50$				via w, ∞ via y, 55 via z, 50
W		→ y, $D_w(x)=\infty$ → z, $D_w(x)=50$	→ y, $D_w(x)=51$ → z, $D_w(x)= \infty$		via w, ∞ via y, ∞ via z, 51
Y		→ w, $D_y(x)=53$ → z, $D_y(x)= \infty$		→ w, $D_y(x)= \infty$ → z, $D_y(x)= 52$	via w, 52 via y, 60 via z, 53

c) cut the link between y and z.



Problem 49

The center-based tree for the topology shown in the original figure connects A to C; B to C; E to C; and F to C (all directly). D connects to C via E, and G connects to C via D, E. This center-based tree is different from the minimal spanning tree shown in the figure.