Problem 1)

a. Host A determines inspecting the 8-bit protocol field in an IPv4 header or the next header field in an IPv6 header, which contains a numerical value indicating the appropriate transport layer protocol. I believe it is 6 for TCP and 17 for UDP.
b. Yes, a host can have multiple IPs through multiple networks like WiFi, Ethernet, etc. that each has its own unique IP address. Even a server can have multiple virtual interfaces with different IPs.
c. Session Traversal Utilities for NAT discovers public IPs and ports while Traversal Using Relays around NAT relays traffic through public servers if you can't connect peer-to-peer. For example if you had public IP A and your friend had public IP B, a STUN protocol can be used to discover the IPs and ports.
d. It wouldn't be needed as NAT existed in case of an IP address shortage under IPv4. With IPv6 we go from 2^{32} IPs to 2^{128} IP addresses which significantly reduces the need for NAT. Some firms might still prefer it though for security reasons.

Problem 2)

a. There are 8 subnets: 223.1.1.0, 223.1.2.0, 223.1.3.0, 223.1.4.0, 223.1.7.0, 223.1.8.0, 223.1.9.0, 223.1.10.0
b. DHCP discover, DHCP offer, DHCP request, and DHCP ACK
c. The main issue is that PC0 is now in the 223.1.10.0 subnet but has an IP from 223.1.2.0. This can lead to PC0 attempting to send data through a gateway that is no longer available. DHCP solves this problem by auto assigning PC0 a correct IP for its new subnet.

Problem 3)

If subnet 1 needs to hold 100 interfaces, subnet 2 needs to hold at least 50 interfaces, and subnet 3 needs to hold 25, we have 7, 6, and 5 host bits respectively. The subnet masks are 25, 26, and 27 respectively too. This means we have to allow $2^7 = 128$ spaces for subnet 1, $2^6 = 64$ spaces for subnet 2, and $2^5 = 32$ spaces for subnet 3.

Subnet 1: 10.0.0.0/25
Subnet 2: 10.0.0.128/26
Subnet 3: 10.0.0.192/27

Problem 4)

a.

b.

Problem 5)

Step by step implementation (highlight means current node):

Visited Set: {t}

Row	**Cost**
t:	0
u:	2
v:	4
w:	∞
x:	∞
y:	7
z:	∞

Visited Set: {t, u}

Row	**Cost**
t:	0
u:	2
v:	4 = min(4, 2 + 3)
w:	5 = 2 + 3
x:	∞
y:	7
z:	∞

Visited Set: {t, u, v}

Row	**Cost**
t:	0
u:	2 = min(2, 4 + 3)
v:	4
w:	5 = min(5, 4 + 4)

x:	7 = 3 + 4
y:	7 = min(7, 4 + 8)
z:	∞

Visited Set: {t, u, v, w}

Row	Cost
t:	0
u:	2 = min(2, 5 + 3)
v:	4 = min(4, 5 + 4)
w:	5
x:	7 = min(7, 5 + 6)
y:	7
z:	∞

Visited Set: {t, u, v, w, x}

Row	Cost
t:	0
u:	2
v:	4 = min(4, 7 + 3)
w:	5 = min(5, 7 + 6)
x:	7
y:	7 = min(7, 7 + 7)
z:	15 = 7 + 8

Visited Set: {t, u, v, w, x, y}

Row	Cost
t:	0

u:	2
v:	4 = min(4, 7 + 8)
w:	5
x:	7 = min(7, 7 + 6)
y:	7
z:	15 = min(15, 7 + 12)

Visited Set: {t, u, v, w, x, y, z}

Row	**Cost**
t:	0
u:	2
v:	4
w:	5
x:	7 = min(7, 15 + 8)
y:	7 = min(7, 15 + 12)
z:	15

Problem 6)

a. The initial distance from y => x is 4, distance from z => x is set to 50, but we have the alternatives z => y => x and z => w => y => x which is 7 and 6 respectively, getting us z => x is 6. w => x would be w => y => x and w => z => x which is 5 and 51 respectively, so w => x would be 5. In conclusion,
   w => x is 5
   y => x is 4
   z => x is 6
b. If y => x gets set to 60, w and z will still think that y has a cost of 4 to x so w thinks it can reach x via y in 1 + 60 = 61 while z thinks it can get to x in 1 + 1 + 60 = 62. But then each router thinks another has a shorter path leading to count to infinity problem, thus not being able to resolve the problem within 5 iterations.
c. It can once the router realizes that there is no low-cost path to x anymore and instead will stabilize to y => x being 60 units. This process can take many steps.