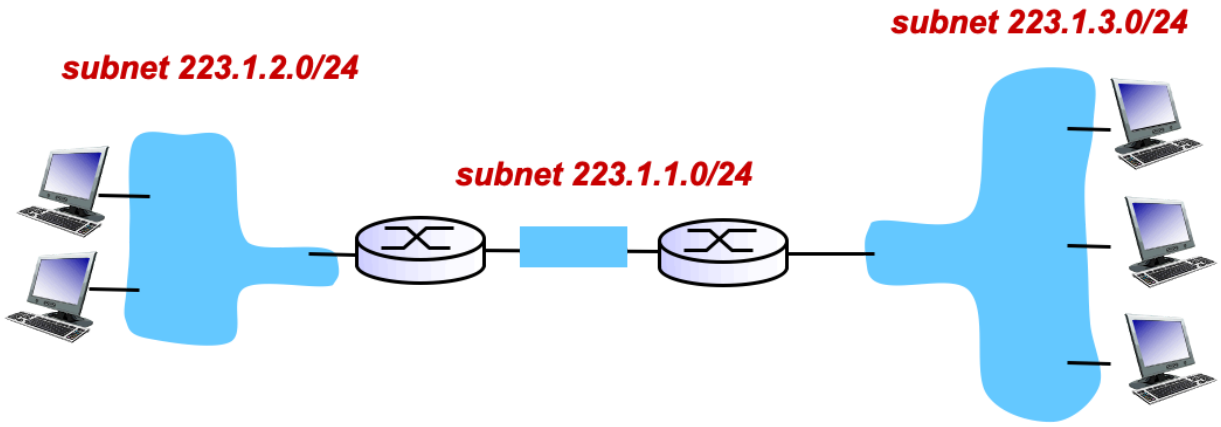
Worksheet Class-16: The Network Layer & IP

Q1.Consider the three subnets in the figure and complete your answers to the following questions.



A) The number of interfaces (in total) in the three subnets combined is ___
B) The maximum number of interfaces in 223.1.2.0/24 network is
C) The maximum number of hosts in 223.1.3.0/29 network is __32-29 =
D) Provide a valid IP address for a host in the 223.1.3.0/29 network:

Q2. Longest prefix matching. Suppose a router uses longest-prefix matching and has the following forwarding table

Destination IP Prefix	Link Interface
10*	1
00*	2
101*	3
001*	4
011*	5
Otherwise	6

Suppose the following datagrams arrived at the router, to which interface will this datagram be forwarded using longest-prefix matching?

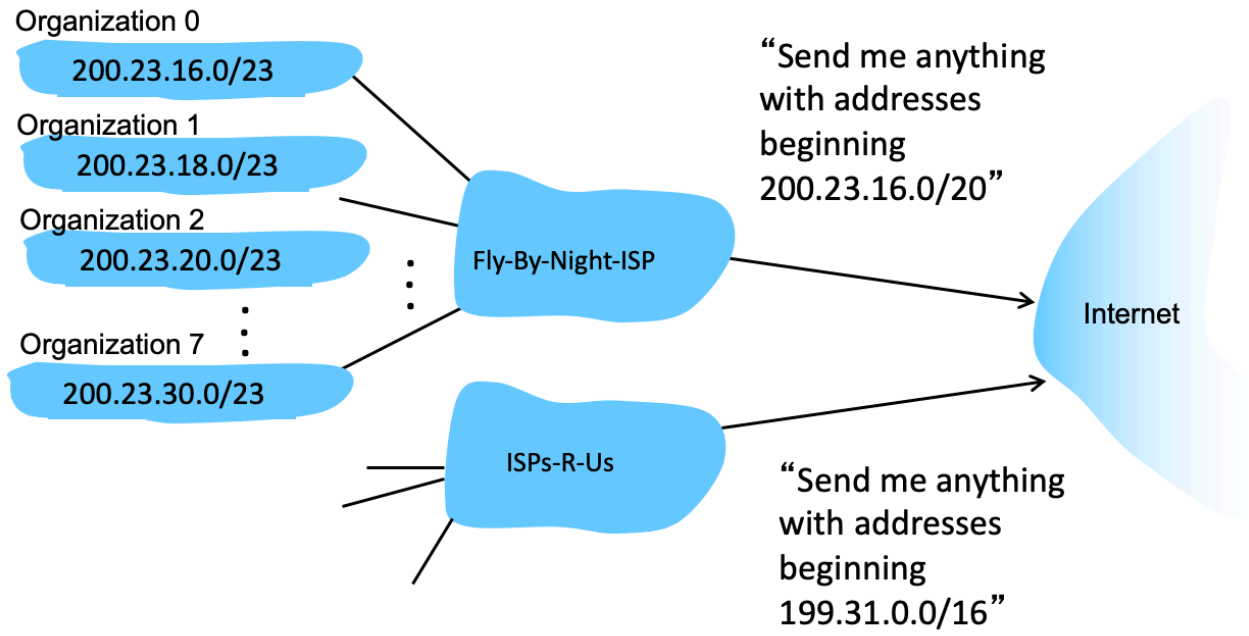
1. destination address 01011110 Interface:
2. destination address 00110001 Interface:
3. destination address 10011001 Interface:

Q3. Why do we give out addresses in CIDR (Class-less addressing) blocks? How many of these statements are true? (Which ones?)

A) It requires fewer resources at routers. More resources
B) It requires fewer resources at end hosts. Not any different
C) It reduces the number of block allocations that need to be managed. False
D) It better utilizes the IP address space. True

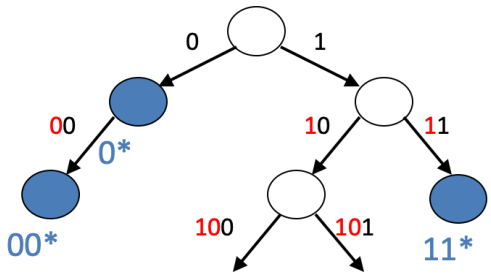
Q4. What should we do if organization 1 in the figure below, decides to switch to ISPs-R-Us? Think about how longest prefix matching works to come up with a solution to this problem.

A) Move 200.23.18.0/23 to ISPs-R-Us (and break up Fly-By-Night's /20 block).
B) Give new addresses to Organization 1 (and force them to change all their addresses).
C) Some other solution



Org1 - ISPs-R-US
200.23.18.0/23

Q5. Instead of a hardware look-up like a TCAM, let's say we used software look-up data structures. What is the time complexity of a look-up for the following data structures, assuming the forwarding table is of size n, where each entry matches on up to p bits.

Forwarding Table Data structure	Time Complexity of a look-up
Algorithm 1 ● Scan the forwarding table one entry at a time ● See if the destination matches the entry ● If so, check the size of the mask for the prefix ● Keep track of the entry with longest-matching prefix	
Store forwarding table as a binary prefix tree, with depth p bits. 	
Store forwarding table as a k-ary prefix tree with depth p/k bits:	