As part of support the routing packets based on destination IPv4 addresses, an operation called a longest prefix match is required. For example, let's assume we want to route a packet destined for 12.3.132.1, and the routing table contains the following data:

Prefix	Mask	Next Hop
12.3.4.0	24	Eth0
12.3.0.0	16	Eth1
12.3.128.0	17	Eth3
12.3.4.5	32	Eth4
12.3.4.2	32	Eth5
15.1.2.89	32	Eth6

We need to determine on which interface we should send the packet. If we look at the address 12.3.132.1, it can be represented in binary as 00001100.00000011.10000100.00000001 . If we look at the above table with a binary representation of the prefixes, it looks like:

Prefix	Mask	Next Hop
b00001100.00000011.00000100.00000000	24	Eth0
b00001100.00000011.00000000.00000000	16	Eth5
b00001100.00000011.10000000.00000000	17	Eth3
b00001100.00000011.00000100.00000101	32	Eth4
b00001100.00000011.00000100.00000010	32	Eth5
b00001111.00000001.00000010.01011001	32	Eth6

Looking at the above, we see that the longest matching prefix is 12.3.128.0 because:

b00001100.00000011.10000100.00000001 and b00001100.00000011.10000000.00000000 have the same first 17 bits, where as 12.3.4.0 has 0 (i.e. the first 24 bits are not identical), 12.3.0.0 has 16 identical bits, 12.3.4.5 has 0 as does 12.3.4.2 and 15.1.2.89.

We would like you to define the data structure to store the list of prefixes (i.e. the above table) that can be searched efficiently (i.e. space and time complexity) to determine the matching output interface over which the packet should be transmitted, along with the matching functions to add a new prefix to the table and obtain the output interface. Your code should be written in C.