1. Why do we use a link for the shortest path computation only if it exists in our database in both directions? What would happen if we used a directed link AB when the link BA does not exist?

My implementation of Djikstra's algorithm is not checking whether edges are symmetrical, because the simulation we are doing has only included symmetrical edges so far. If we did have asymmetrical edges, then this could result in bad behavior if we try to send an acknowledgement packet down the same route as the original packet. I'm not exactly sure what the larger issue is.

2. Does your routing algorithm produce symmetric routes (that follow the same path from X to Y in reverse when going from Y to X)? Why or why not?

No. Nodes search essentially by alphabetical order for lowest cost neighbors in my implementation of djikstra's algorithm, and because all edge weights are 1, this means that any given node will simply just be biased to pick whatever neighbor has a link to the endpoint of lowest weight and first in alphabetical order.

3. What would happen if a node advertised itself as having neighbors, but never forwarded packets? How might you modify your implementation to deal with this Case?

In order to even realize that packets are being lost at a random point in the network, we would need to have acknowledgement packets being sent back. If we had information about packet loss, we could try to have nodes run some statistical analysis and include it in their path-finding for djikstras algorithm, where paths which have lower percentage chances of reaching the target are given higher weight.

4. What happens if link state packets are lost or corrupted?

If the protocol header on one of my link state packets is corrupted, then there will be undefined behavior. If the destination header is corrupted, then the packet will be sent to a new destination. If the packet is outright lost, absolutely nothing happens and nobody remembers its loss.

5. What would happen if a node alternated between advertising and withdrawing a neighbor, every few milliseconds? How might you modify your implementation to deal with this Case?

If a node continuously changes a neighbor value, then there will be exactly a 50% chance that it is used during routing updates, and a 50% chance that it is ignored. I'm not quite sure that this case would be highly problematic, as long as the packets actually reach their destination. If the packets don't reach their destination, then we would need to use acknowledgement packets to prevent fake neighbors being advertised.