Problem 1

a) This program emulates the *ls*(1) function, and then copies itself to new directories. The first line runs *ls* on the given arguments, which are specified by "$@". It then stores the exit code (specified by $?) in a variable that it returns at the end of the program. This functionality just emulates *ls*. The next part is responsible for replicating the program to other folders. The *DIRS* variable stores each separate path on the *PATH* variable, which is done by replacing each ":" (used to separate paths) with a newline character by using *tr*(1). The program then loops through each directory in the path, and then copies the script itself (specified by *BASH_SOURCE* to a file named *ls* within that directory. From there, it makes the file executable and readable for everyone by calling *chmod*(1). So, the *ls* script has been run and then copied to every directory in Jake's *PATH*. This could happen if the script was run within Jake's filesystem, then the program will be copied to his personal *bin* directory because it's on his path. Because the file is named *ls*, it's possible that Jake just ran it by mistake while thinking he was running the standard *ls*.

b) The author could write a similar script for the *which*(1) function. The main defense against this attack is to check that the user is running the correct *ls* function, by running *which ls*. If we write a script called *which*, if the argument passed in is *ls*, then the function should *echo /bin/ls* (the address of the "real" *ls* command). Otherwise, the script should just call *which* $@ and return its error code on exit so that the user will be less likely notice it's not the real *which* script.

Problem 2

a) It is not possible to come up with an algorithm that determines if P will execute the body of the virus. Suppose for the sake of contradiction that such an algorithm exists, and we can call that algorithm *executes*(program P), and it will return True if P executes the virus, and return false otherwise. In the same style as the halting problem, we can write a function *prog*(). *prog*() will check if *executes(prog())* returns true. If it doesn't, then the virus will be executed, otherwise the function will return. This function creates a paradox; if *executes(prog())* returns false, then the virus will be executed, making the output of *executes* incorrect. If it returns true, then the virus won't be executed, again making the output incorrect. Thus, no such algorithm can exist.

b) Such an algorithm exists and runs in $O(n^2)$ time. We know what C is and do not know what K is, but we know that if K exists, it's stored somewhere in the code. So, we need to loop through every possible potential value for K, and every potential location of the virus. We can then use T, which is $K++K++\ldots++K$, to find $O' \oplus T$, and compare this to C. If they're equal, then the virus has infected the program, otherwise, we try the next guess. As there number of potential values for K is linear with regards to the length of P, and the number of potential locations for C is also linear with regards to the length, this program runs in $O(n^2)$ time, where n is the length of P.

Problem 3

a) We want B to send an ARP reply to A that says that IP address 192.168.1.4 is at MAC address 33:33:33:33:33:33. This would poison A's ARP cache, and cause A to send packets to B which were originally intended for C.

b) When B receives packets from A that were intended for C, it should forward these packets to C so that C doesn't realize that eavesdropping is occurring. To make it seem like the packets came from A, the src attribute of the IP header should use A's src IP address, not B's.

c) We use the same process. B sends an ARP reply to C that says that IP address 192.168.1.2 is at MAC address 33:33:33:33:33:33. This means that any data C sends "to A" will instead go the B. From there, B should forward these packets to A, while modifying the IP header so that the src attribute uses C's IP address, so that A thinks the packet came from C.

d) We use the same process. B sends an ARP reply to A that says that IP address 128.148.32.12 is located at MAC address 33:33:33:33:33:33. When B receives packets from A, it will forward these packets the actual destination IP address. B will also spoof the router by poisoning its cache. B will send an ARP reply to the router which says that IP address 192.168.1.2 is located at MAC address 33:33:33:33:33:33. Thus, when the router receives packets intended for A from the external node, it will send those packets to B. B should forward these packets to host A, while ensuring the source address in the IP header refers to the external node, and not host B.

Problem 4

a) I am in favor of this proposal for small updates that fix security bugs which affect other users, but not for all security fixes. Especially with operating systems, not all programs are compatible with all updates, so by forcing an update, you can make life very difficult for the user. However, if the security risk has significant chance to affect other users, then the update should be mandated, since individual users shouldn't be trusted to weigh the impacts of their decisions on others.

b) I think companies should be liable regardless of their cybersecurity best practices. It's not clear that cybersecurity best practices will always be exactly the same, and in general, laws like this will encourage companies to do the absolute minimum. If there's no minimum standards for exemption, then companies will have a major financial incentive to use best security practices.

c) (i) The CSO article argues that the cascading effects of breaches creates moral hazard for companies. They can save money on the effects of breaches, even though those breaches can effect their customers, not just themselves. The article argues that paying for insurance encourages teams not to follow good security practices.
(ii) I believe cybersecurity insurance is indeed a barrier to investment, because of the moral hazard issue that the article raises. I think that it is reasonable for insurance to cover losses that a company suffers, except when those losses are from lawsuits by affected third-parties. That is, if a companies bad practices lead to another company being affected, the first company should still be fully liable for their mistakes. Legislation is essential for guaranteeing that companies are compelled to invest in cybersecurity, and for prohibiting insurance form being a barrier.

d) I wouldn't change the author's proposal. I think intentionally inflicting damage should be a criminal matter. I also think it's important to shield those who provide open-source software who don't necessarily have the resources that a large company would. Finally, I think companies have to be liable for the damage they inflict. Companies need a major financial incentive to invest in cybersecurity, and the risk of liability is such a financial incentive.