Checksum

* Sometimes information can get garbled as it travels on the internet – kind of like a letter that got wet before it gets to its destination.
* We do something cool with computer information to make sure that it gets to its destination safe and sound
* We do something called a checksum
* Computer information is stored and sent places as a string of 0’s and 1’s
* They make up a number system called binary
* If you add all the segments of 0’s and 1’s together and save the number you get, then do the same addition once the information has gotten to its destination, you can check to see if it’s garbled.
* It’s kind of like this – suppose you send the numbers 4,5,6 to somebody else, along with the sum, 15.
* If any of the numbers changed, and you tried to add them together again, you wouldn’t get 15, and you’d know that the information was wrong.
* Then you could ask to send it again.
* You add binary like this: 0 + 1 = 1, 0 + 0 = 0, and 1 + 1 = 1, carry a 1
* When doing checksums, we can ignore the last carry if there is one
* Now you try! If the numbers added together match the sum below them, hit the check button. If the sum and the checksum don’t match, hit the X button.

Pathfinder

* Before information can travel on the internet from one computer to another, a machine called a router chooses the paths that are the fastest or the shortest
* Before we can send a package, we have to figure out how to get it to its destination
* Each road on the map has a different number – this represents how fast it is to travel along the road. Higher numbers mean more traffic – even a short looking road might take a lot more time to get across.
* Your job is to find the best path from one post office to the other – look at all the different road options and tap the ones that connect to make the shortest path.

Segmentation

* Information travels along a network in packets
* Sometimes you want to send more information than will fit in one packet
* So you split it up into more than one – they are called segments
* Sometimes as segments are getting sent along the network, they get out of order
* But if they are out of order, the information won’t make sense once it has reached its destination
* Each segment has a kind of label on it that tells the destination which segment it is
* Then the destination can ignore any segments that aren’t in order and wait for the sender to resend the information in that segment
* We will practice this concept with packages – we’ve got shipments with too much stuff to fit in one box. They have been separated into packages.
* Your job is to drag the packages into the correct mailboxes, but only if they are in the right order