Problem 3

Rong Kang Chew (Ron)
https://github.com/hooplapunta/airports-zipcode

My approach to Problem 3 was to break down the problem into 2 parts:

1. To acquire a list of all airports in the world into a parse-able list or database.
2. To get the postal code for each of those airports.

I know the first problem has probably been solved already. I found a number of GitHub repositories that had airport information in a JSON file database (https://github.com/mwgg/Airports) and decided not to "re-invent the wheel". Should this database not have been available, I would have written my own scraper to lookup websites like https://airports-list.com/icao-code.

Looking up the Google Maps was more problematic than I first thought it to be. My initial approach was to use the latlon coordinates provided by the input database to reverse geo-code and get the postal code from Google. The problem is that not all of these coordinates land in a spot that Google Maps can cough out a postal code for.

I therefore had to take a multi-step approach for fallbacks in case the first approach didn't work:

1. Reverse geocode the initial latlon provided by input, and check if there is a postal code found.
2. Geocode the name provided by input, check if there is a postal code found.
3. Reverse geocode the first entry's latlon in (2), check if there is a postal code found. (i.e in case the first latlon was wrong or off by a few, e.g. located on the airstrip)
4. If there is no postal code found at this point, skip the entry.

Quite a number of airports required going to step 3, including our own Changi Airport, WSSS.
The solution I wrote runs on **node.js**. I was initially planning to write a React frontend to view this data and perhaps trigger the lookup process. However, I ran out of time to complete this.

I unfortunately had to write this solution while travelling on a volunteer trip Habitat for Humanity to Laredo, Texas. If I was given more time, I would have written a more robust solution without callback hell (3 nested callbacks) and that was throttled a little better. Running this solution without throttling would overwhelm the TCP/IP buffer and is also prone to timeouts. Looking up the 40,000 some airports with throttling can take up to 30 minutes.

I might also look for alternative solutions to looking up the postal codes - there are some websites that are dedicated to postal codes and might be able to provide that information. These websites however do not have an API and would similarly require a webscraper or scripter to trigger the lookups.

Output data
You can find the output data in ./output/airportsPostalCode.json.
The data is formatted as a JSON object (i.e. dictionary) of a ICAO airport code to the postal code and the search level it was found at (see above for explanation).
You can find the original input data in ./output/airports.json.

Offsite Issues
I am currently based offsite in Laredo, TX on a volunteer trip with Habitat for Humanity, where 4G connection is a little spotty. Therefore a large number of my requests timed out and the output

database incomplete. You may want to re-run **> node icaoToZipcode.js** on your own machine to rebuild the database. My key is included in this file for your convenience - please do not attempt to run it multiple times if there is an error. In an actual production environment, I would secure this key in a proper keystore or provisioning system in CI.