Names of team members: Jet Tom, Kevin Mooney

Step 1: Download the compressed table DB_091803_v1.txt and test input file: IPlist.txt. Upload them to your Colab space using the code below.

```
from google.colab import files
uploaded = files.upload()
DB = next(iter(uploaded))
print(DB, "uploaded")
```

Choose Files DB_091803_v1.txt

- **DB_091803_v1.txt**(text/plain) - 3410045 bytes, last modified: 2/9/2023 - 100% done

```
Saving DB_091803_v1.txt to DB_091803_v1.txt
DB_091803_v1.txt uploaded
```

```
uploaded = files.upload()
IP_LIST = next(iter(uploaded))
print(IP_LIST, "uploaded")
```

Choose Files IPlist.txt

- **IPlist.txt**(text/plain) - 149 bytes, last modified: 2/9/2023 - 100% done

```
Saving IPlist.txt to IPlist.txt
IPlist.txt uploaded
```

Step 2: Develop the IP2AS tool - This tool maps an IP address to an AS. It uses static table address prefix to AS number collected from whois DB and BGP tables. This table is stored in a file and should be given to the tool as a parameter. It will perform longest prefix matching and will map the IP to an AS number. The tool should print out the longest prefix that the IP address is matched to, and the corresponding AS number.

Steps:

1. Put the set of IP addresses you want to map to ASes into the `<IP file>`. You can list one IP address per line.

   For example, look at IPlist.txt file, which contains the following:

   ```
   169.237.33.90
   208.30.172.70
   ```

2. The `<DB file>` has data about which address block belongs to a particular AS (look at `DB_091803_v1.txt` file, for example). The `<DB file>` is constructed based on IRR database and BGP routing table.

3. Run ip2as and specify the and

   For example, if you run your code, the output should look like the following:

   ```
    169.237.0.0/16  1852 169.237.33.90
    208.0.0.0/11  1239 208.30.172.70
   ```

```
# The code should assume that the variables db_filename and input_filename
# contain the filenames of the database and inputs respectively.

#This program (i) takes an IP address as an input, (ii) performs the longest prefix match on a compressed version
#of the routing table, and (iii) look up & output the home AS number for that IP address.
#It is assumed that the DB file contains an ip mask and asn in a single line with a whitespace character
#between them and is formatted in such order: ip, mask, asn

#reading IPlist.txt file
IP_address = open(IP_LIST, "r")
ip_content = IP_address.read()

#converting IP address to binary for IPlist.txt

#variables:

#IP_addr = individual ip addresses from ip_content parsed at new line character

#ip_bin = list of ip addresses from IPlist.txt converted to binary in the form "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

IP_addr = ip_content.split("\n")
IP_addr.pop()
```

```
ip_bin = []
for ip in IP_addr:
  token = ip.split(".")
  ip_bin.append('{0:08b}{1:08b}{2:08b}{3:08b}'.format(int(token[0]),int(token[1]),int(token[2]),int(token[3])))

#reading DB file

#variables:

#db_tok = list of all entries of db file (unparsed)

#db_ip_bin = list that will store binary ip addresses of db file
db_file = open(DB, "r")
db_content = db_file.read()
db_tok = db_content.split("\n")
db_tok.pop()
db_ip_bin = []
int_db_routing_prefix = []

#converting database IP addresses into binary for prefix matching
#error message if ip address is invalid, and does not append result to list of converted binary numbers

#variables:
#line = list of entries parsed at " " character

#line_ipaddr = first element of line, split apart by "."

#routing_prefix_typeInt = second element of line (subnet mask) converted to #type int

#db_to_bin = line_ipaddr converted to binary in the form "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
line_count = 0
for i in db_tok:
  line = i.split(" ")
  line_ipaddr = line[0].split(".")
  routing_prefix_typeInt = int(line[1])
  db_to_bin = ('{0:08b}{1:08b}{2:08b}{3:08b}'.format(int(line_ipaddr[0]),int(line_ipaddr[1]),int(line_ipaddr[2]),int(line_ipaddr[3
  if(len(db_to_bin) != 32):
    print("Error: Invalid IP address at line", line_count, ", IP address:", line[0])
    db_tok.pop(line_count)
  db_ip_bin.append(db_to_bin)
  int_db_routing_prefix.append(routing_prefix_typeInt)
  line_count += 1

#perform longest prefix matching
#match if all digits of subnet mask is matched entirely

#variables:

#curr_ip = current binary ip address in IPlist.txt

#IPlist_element = counter to iterate through IPlist.txt

IPlist_element = 0
for ip in ip_bin:
  curr_ip = ip
  long_counter = -1
  long_index = -1
  for index in range(0, len(db_ip_bin)):
    counter = 0
    while curr_ip[counter] == db_ip_bin[index][counter] and counter < int_db_routing_prefix[index]:
      counter += 1
      if counter > long_counter and counter == int_db_routing_prefix[index]:
        long_counter = counter
        long_index = index
  print('/'.join(db_tok[long_index].split(" ", 1)), IP_addr[IPlist_element])
  IPlist_element += 1
```

```
Error: Invalid IP address at line 19788 , IP address: 66.5459.101.0
Error: Invalid IP address at line 82730 , IP address: 202.13131.32.0
Error: Invalid IP address at line 139922 , IP address: 211211.29.0.0
12.105.69.144/28 15314 12.105.69.152
12.125.142.16/30 6402 12.125.142.19
57.0.208.244/30 6085 57.0.208.245
208.148.84.0/30 4293 208.148.84.3
208.148.84.0/24 4293 208.148.84.16
208.152.160.64/27 5003 208.152.160.79
192.65.205.248/29 5400 192.65.205.250
194.191.154.64/26 2686 194.191.154.80
```

```
        199.14.71.0/24 1239 199.14.70.79
        199.14.70.0/24 1239 199.14.70.79
```

✓ 3s completed at 4:43 PM