Names of team members: Aroop Biswal, Geoffrey Mohn

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 No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
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 No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
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
```

```
# Takes in a string formatted as int and returns a string formatted as binary
def toBin(input):
  # split string by the full stop
  split_list = input.split(".")
  output = ""
  for i in range(len(split_list)):
    # Convert each section into an integer
    split_list[i] = int(split_list[i])
    # Convert integer into 8 bit binary number https://stackoverflow.com/questions/1395356/how-can-i-make-bin30-return-00011110-instead-of-0b
    newChunk = '{0:08b}'.format(split_list[i])
    output += newChunk
  return output
```

```

```

```
# The code should assume that the variables db_filename and input_filename
# contain the filenames of the database and inputs respectively.
#
# Model output is provided for you.
#
# Your code with documentation
```

```
db_filename = DB
input_filename = IP_LIST

# Open IP list and Database
read_db = open(db_filename,'r')
read_ip = open(input_filename,'r')


# Iterate through each line of IP list
while read_ip:
  curr_ip = read_ip.readline()

  # Set the offset of database to the beginning of the file
  read_db.seek(0)

  # Set the longest common prefix variables
  lcp_ip = ""
  lcp_length = 0
  lcp_bit = 0
  lcp_asn = ""
  ip_len = len(curr_ip)

  # Break loop when at EOF
  if(curr_ip == ""):
    break

  # Iterate through the database with the current read IP
  while read_db:
    db_line = read_db.readline()
    # If EOF print out result of lcp
    if (db_line == ""):
      print(str(lcp_ip)+"/"+str(lcp_bit)+" "+str(lcp_asn)+ " "+str(curr_ip),end="")
      break
    # parse database line
    parsed_str = str.split(db_line)
    db_ip = parsed_str[0]
    db_bit = parsed_str[1]
    db_asn = parsed_str[2]
    # count length of prefix length
    prefix_len = 0
    # Convert both ip line and database line into binary
    curr_ip_bin = toBin(curr_ip)
    db_ip_bin = toBin(db_ip)

    # Iterate through length binary number of current ip
    for i in range(len(curr_ip_bin)):

      # If we have checked the number of bits distinguished in the database then check to update the lcp.
      if(i == int(db_bit)):
        # If the iterator is longer than the currently stored lcp_length then update all variables of lcp
        if(i > lcp_length-1):
          lcp_length = prefix_len
          lcp_ip = db_ip
          lcp_asn = db_asn
          lcp_bit = db_bit
          break
      # If match is found between ip and db bit then increment prefix_len.
      if curr_ip_bin[i] == db_ip_bin[i]:

        prefix_len += 1
      # break to the next database's line.
      else:
        break


read_db.close()
read_ip.close()
```

```
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 199.14.71.79
199.14.70.0/24 1239 199.14.70.79
```


✓ 15s completed at 6:21 PM