Automated IP subnetting

Question 1: Automated IP subnetting

In this assignment, you will implement a program that accepts a network's address and list of required subnet sizes and computes the address of each subnet using the subnetting approach we discussed in class.

Input format

The input is a JSON string consisting of the following keys:

- **network_addr**: The address of the network to be divided into smaller networks.
- **netmask**: Netmask of the above network address.
- **subnets**: A dictionary of subnets required. The keys are the subnet IDs and values are the number of usable hosts on the subnet.

Below is a sample JSON string supplied to your tool. This string is displayed across multiple lines for readability: the actual string will be contained in 1 single line terminated by newline character('\n').

```
{
  "network_addr": "192.168.128.0",
  "subnets": {"1": 13, "2": 11, "3": 12},
  "netmask": "255.255.224.0"
}
```

In the above input, the network 192.168.128.0/19 should be divided into 3 networks of sizes 13, 11 and 12 respectively. In this case, it is possible to divide but there will be test cases where it is not possible to divide the network as per the required specification.

Output format

The output of your tool should be a JSON string with the following keys:

- **success**: Boolean value indicating whether the subnetting was successful and all subnets were created as per requirement.
- **subnets**: A dictionary with keys as network IDs and values as another dictionary with the below keys. This needs to be present only if subnetting is possible.
  - **network_addr**: The network address of the subnet.
  - **netmask**: The netmask of the subnet.
  - **start_addr**: The first address of subnet.
  - **end_addr**: The last address of subnet.
  - **total_host_count**: The total number of usable hosts in the network.

For the input described in previous section, a possible output is the following JSON string. This string is displayed across multiple lines for readability: the actual JSON string should be printed to stdout in 1 single line terminated by newline character('\n').

```
{
  "success": true,
  "subnets":
    {
      "1":
      {
        "network_addr": "192.168.128.0",
        "netmask": "255.255.255.240",
        "start_addr": "192.168.128.0",
        "end_addr": "192.168.128.15",
        "total_host_count": 14
      },
      "2":
      {
        "network_addr": "192.168.128.32",
        "netmask": "255.255.255.240",
        "start_addr": "192.168.128.32",
        "end_addr": "192.168.128.47",
        "total_host_count": 14
      },
      "3":
      {
        "network_addr": "192.168.128.16",
        "netmask": "255.255.255.240",
        "start_addr": "192.168.128.16",
        "end_addr": "192.168.128.31",
        "total_host_count": 14
      }
    }
}
```


How we will test your program

We will pass the name of a file with the input string in format as described above. You can assume that the file already exists before your program is invoked. A sample invocation is shown below:

```
$ python2 server.py subnets.json
```

In above invocation, your tool must read the JSON string present in the file *subnets.json* and print the result as a JSON string, as described above. Don't assume the name of the file: use it from the argument passed to your program.