Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

LAB DA – 4

**Name : ** Gabani Vandit Sureshbhai

**Reg.No : ** 20BCI0090

**Subject : ** CSE 1004 – Network and Communication

**Faculty : ** SALEEM DURAI M.A

Topic : IP Addressing

1. IP Addressing
2. Subnetting

*LAB: Assignment: 4 *

① IP Addressing: → IPv4
→ IPv6

→ To identify type of IP Address:

1. Take an input string containing the IP Address.

2. Split the string using "." as a delimiter.

3. Now after that check number of resulting substring. If number of resulting substring is 4 and all substring values are 0 ≤ value ≤ 255 than it is IPv4.

4. If total number of values are 8 then it is IPv6 type of IP Address.

5. Now if number of substrings is neither 4 nor 8, then the input string is not a valid IP Address.

```
char * token = strtok ( ip-address, "." );
int count =0;
while (token! =null)
{
   count ++;
   token = strtok (NULL, ".");
}
if (count ==4)
{
   for (int i =0; i< 3 ;i++)
   {
      if (valueof (token[i]) >=0 && <= 255)
      {
         continue;
      }
      else
      {
         printf (" Invalid IPv4 Address ");
   }  }
```

```
        printf (" IPV4 Address");
}
elseif ( count = = 8)
{
        printf( " IPv6 Address");
}
else {
        printf( " Invalid IP Address");
}
```

→ Class of IPv4:

1. from the token / splitted string parse the first substring and convert its value from string to integer.

1 - 126	A
128 - 191	B
192 - 223	C
224 - 239	D
240 - 255	E

```
char * token = strtok ( ipv4_add , "." );
int first = atoi (token);
→ if ( first ≥ 1 && first ≤ 126)
printf("Class A ");
elseif ( first > = 128 && first <=191)
        printf( "class B" );
elseif (first >=192 && first< =223)
        printf ("class C");
else if ( first > = 224 && first <=239)
        prinf ( "class D");
else if ( first >=240 && first <=255)
        printf ( "class E");
```

```
else
        printf(" Invalid  IPv4 address");
```

* **Input & Output:**

① 170.32.0.64

This is IPv4 address

class : B

② Enter the IP Address :

192.0.290.0

Not a valid IPv4 Address.

③ Enter the IP Address :

12AB : 0001 : 456F : AABC : 1234 : 1CBA : AAFF : 0020

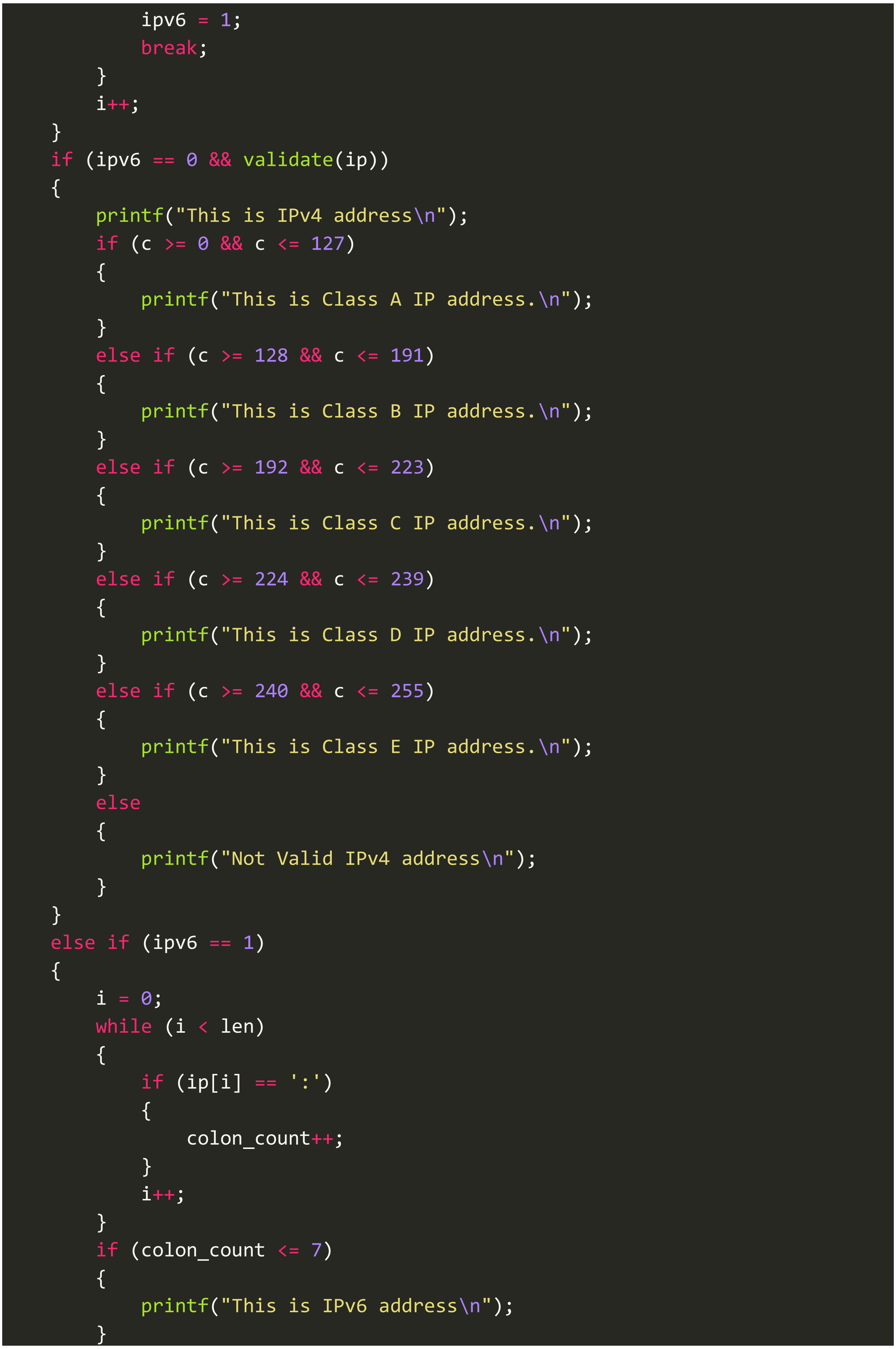
This is IPv6 Address.

Vandit Gabani
20BCI0090

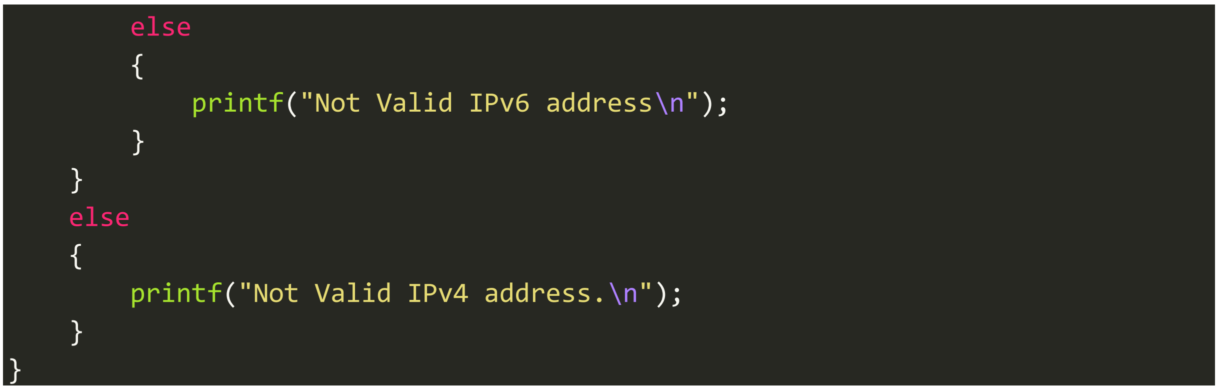
1.IP Addressing:

Code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int c;
int validate(char *ip)
{
    int i, num, check = 0, dots = 0;
    char *ptr;
    if (ip == NULL)
        return 0;
    ptr = strtok(ip, ".");
    if (ptr == NULL)
        return 0;
    while (ptr)
    {
        num = atoi(ptr);
        if (num >= 0 && num <= 255)
        {
            if (check == 0)
            {
                c = num;
                check = 1;
            }
            ptr = strtok(NULL, ".");
            if (ptr != NULL)
                dots++;
        }
        else
            return 0;
    }
    if (dots != 3)
        return 0;
    return 1;
}
int main()
{
    char ip[40];
    int len, i = 0, ipv6 = 0, colon_count = 0;
    printf("Enter the IP address:\n");
    scanf("%s", ip);
    len = strlen(ip);
    while (i < len)
    {
        if (ip[i] == ':')
        {
```



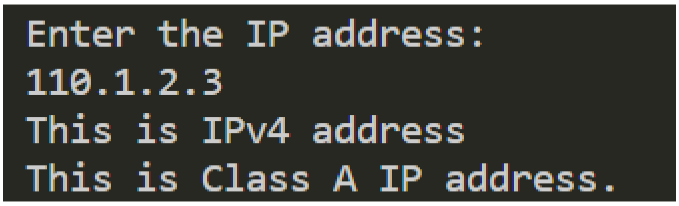
```
            ipv6 = 1;
            break;
        }
        i++;
    }
    if (ipv6 == 0 && validate(ip))
    {
        printf("This is IPv4 address\n");
        if (c >= 0 && c <= 127)
        {
            printf("This is Class A IP address.\n");
        }
        else if (c >= 128 && c <= 191)
        {
            printf("This is Class B IP address.\n");
        }
        else if (c >= 192 && c <= 223)
        {
            printf("This is Class C IP address.\n");
        }
        else if (c >= 224 && c <= 239)
        {
            printf("This is Class D IP address.\n");
        }
        else if (c >= 240 && c <= 255)
        {
            printf("This is Class E IP address.\n");
        }
        else
        {
            printf("Not Valid IPv4 address\n");
        }
    }
    else if (ipv6 == 1)
    {
        i = 0;
        while (i < len)
        {
            if (ip[i] == ':')
            {
                colon_count++;
            }
            i++;
        }
        if (colon_count <= 7)
        {
            printf("This is IPv6 address\n");
        }
```



```
        else
        {
            printf("Not Valid IPv6 address\n");
        }
    }
    else
    {
        printf("Not Valid IPv4 address.\n");
    }
}
```

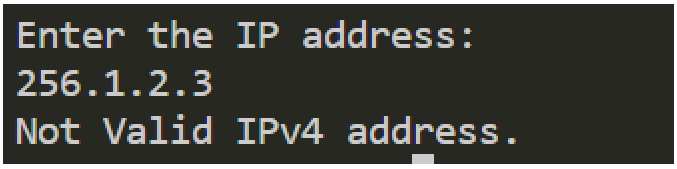
Input – Output:

Valid Ipv4:



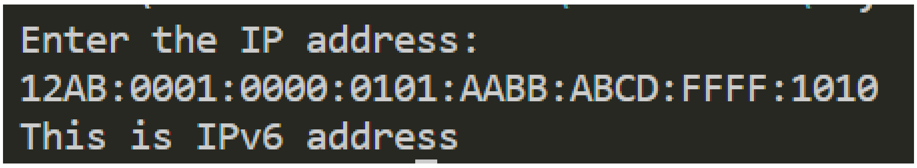
```
Enter the IP address:
110.1.2.3
This is IPv4 address
This is Class A IP address.
```

Invalid Ipv4:



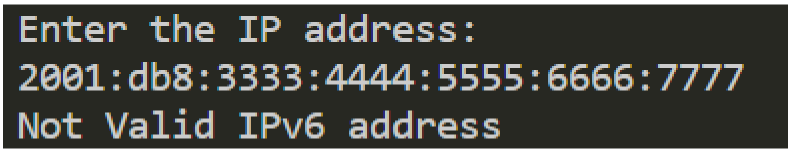
```
Enter the IP address:
256.1.2.3
Not Valid IPv4 address.
```

Ipv6:



```
Enter the IP address:
12AB:0001:0000:0101:AABB:ABCD:FFFF:1010
This is IPv6 address
```

Invalid Ipv6:



```
Enter the IP address:
2001:db8:3333:4444:5555:6666:7777
Not Valid IPv6 address
```

② subnetting:

* Algorithm:

1. Take the input of IP Address from the user and also the number of division or in how many parts the network should be divided from the user.

2. Based on the division try to find out the total number of host bits.

3. Now calculate the subnet mask by setting first n bits to 1 and remaining bits to 0 where n is number of subnet bits.

4. Calculate the number of subnets and hosts per subnet:

   a. Calculate the number of subnet by raising 2 to the number of subnet bits.

   → Calculate the number of hosts per subnet by raising 2 to the number of host bits.

5. Try to determine network address and broadcast addrese.

→ perform AND operation of each octet from the given IP Address and corresponding subnet mask.

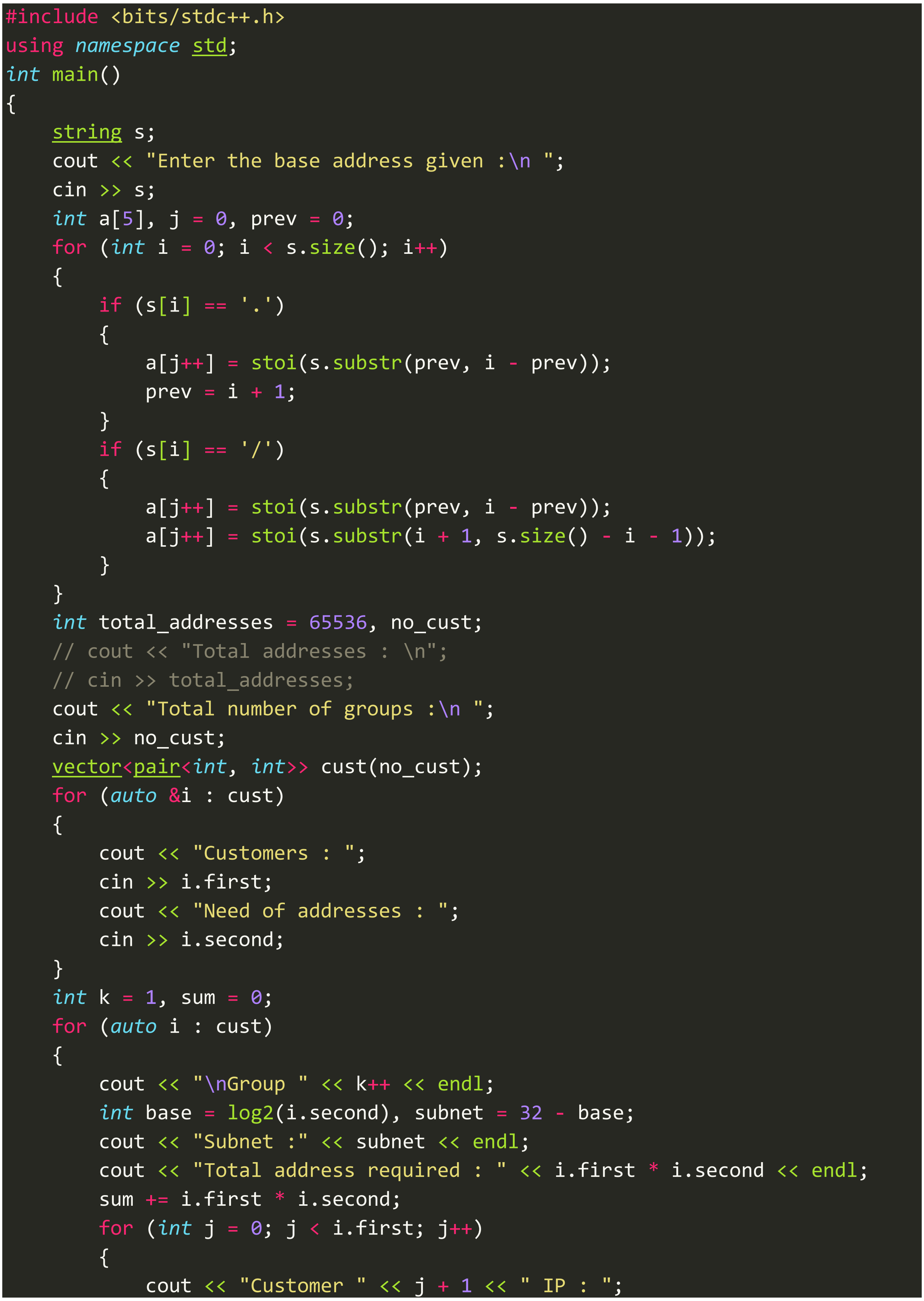
→ perform OR (bitwise) operation on the network address and the bitwise complement of subnet mask and calculate the broadcast address.

6. Calculate the range of usable host addresses for each subnet:

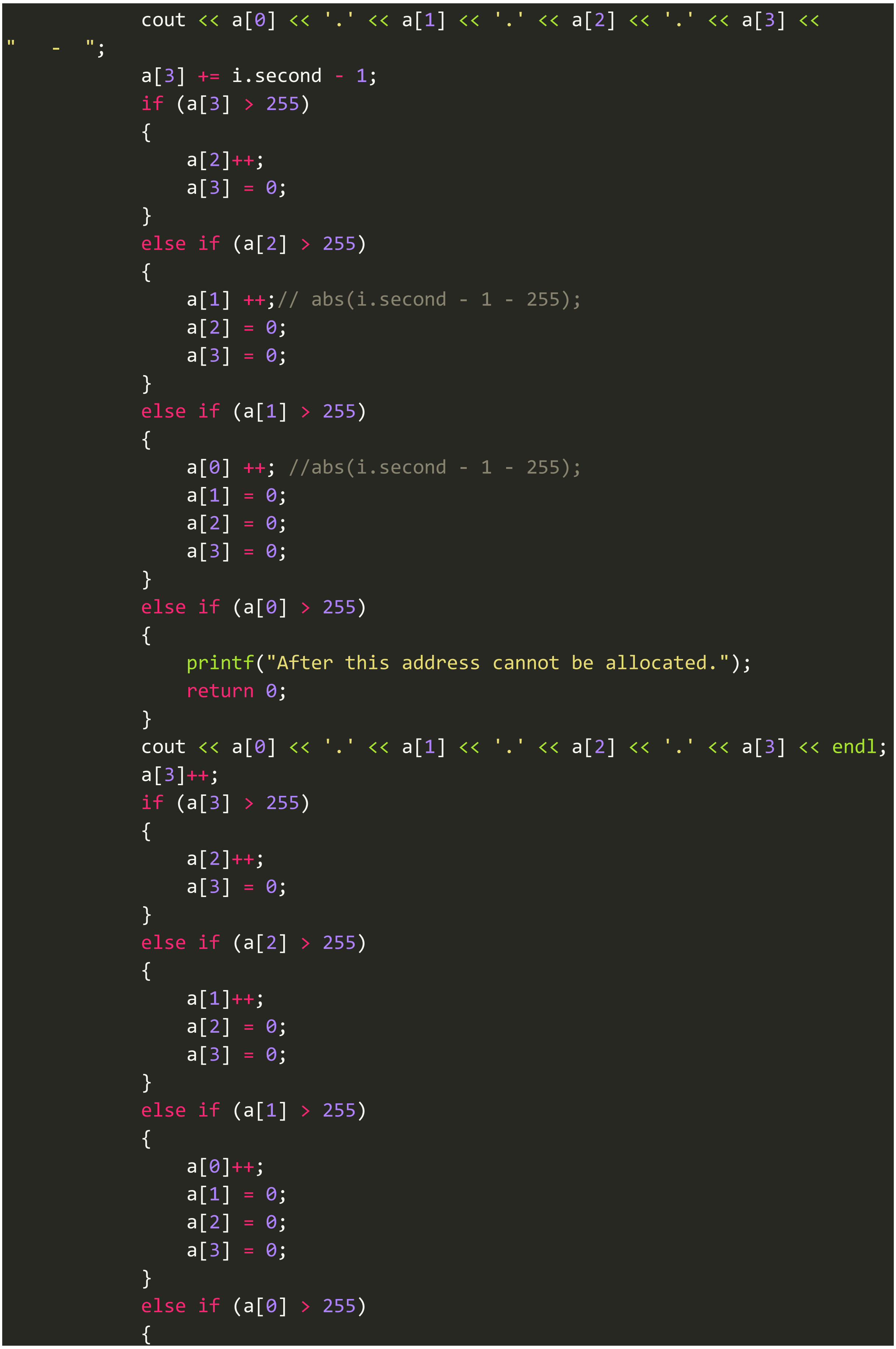
→ substract 2 from total hosts for each subnet.

2.Subnetting:

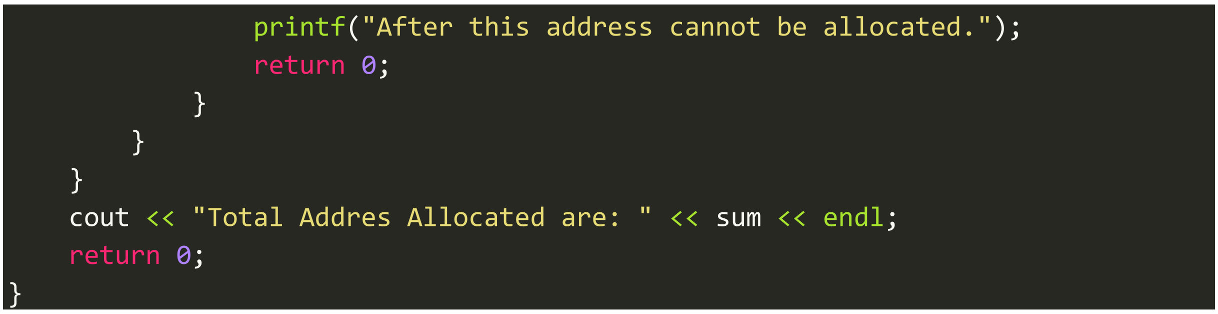
Code:



```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string s;
    cout << "Enter the base address given :\n ";
    cin >> s;
    int a[5], j = 0, prev = 0;
    for (int i = 0; i < s.size(); i++)
    {
        if (s[i] == '.')
        {
            a[j++] = stoi(s.substr(prev, i - prev));
            prev = i + 1;
        }
        if (s[i] == '/')
        {
            a[j++] = stoi(s.substr(prev, i - prev));
            a[j++] = stoi(s.substr(i + 1, s.size() - i - 1));
        }
    }
    int total_addresses = 65536, no_cust;
    // cout << "Total addresses : \n";
    // cin >> total_addresses;
    cout << "Total number of groups :\n ";
    cin >> no_cust;
    vector<pair<int, int>> cust(no_cust);
    for (auto &i : cust)
    {
        cout << "Customers : ";
        cin >> i.first;
        cout << "Need of addresses : ";
        cin >> i.second;
    }
    int k = 1, sum = 0;
    for (auto i : cust)
    {
        cout << "\nGroup " << k++ << endl;
        int base = log2(i.second), subnet = 32 - base;
        cout << "Subnet :" << subnet << endl;
        cout << "Total address required : " << i.first * i.second << endl;
        sum += i.first * i.second;
        for (int j = 0; j < i.first; j++)
        {
            cout << "Customer " << j + 1 << " IP : ";
```

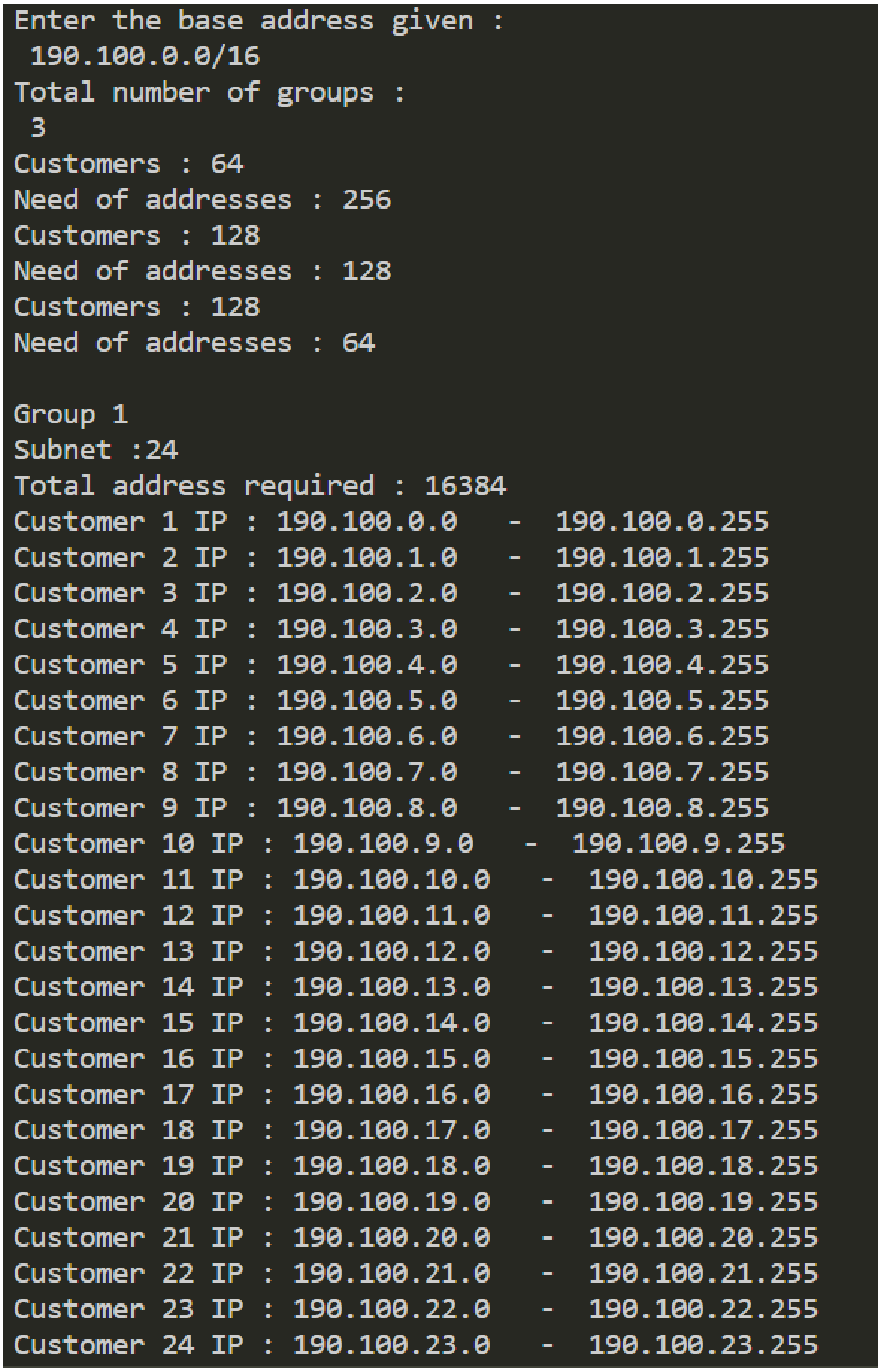


```
            cout << a[0] << '.' << a[1] << '.' << a[2] << '.' << a[3] <<
"   -  ";
            a[3] += i.second - 1;
            if (a[3] > 255)
            {
                a[2]++;
                a[3] = 0;
            }
            else if (a[2] > 255)
            {
                a[1] ++;// abs(i.second - 1 - 255);
                a[2] = 0;
                a[3] = 0;
            }
            else if (a[1] > 255)
            {
                a[0] ++; //abs(i.second - 1 - 255);
                a[1] = 0;
                a[2] = 0;
                a[3] = 0;
            }
            else if (a[0] > 255)
            {
                printf("After this address cannot be allocated.");
                return 0;
            }
            cout << a[0] << '.' << a[1] << '.' << a[2] << '.' << a[3] << endl;
            a[3]++;
            if (a[3] > 255)
            {
                a[2]++;
                a[3] = 0;
            }
            else if (a[2] > 255)
            {
                a[1]++;
                a[2] = 0;
                a[3] = 0;
            }
            else if (a[1] > 255)
            {
                a[0]++;
                a[1] = 0;
                a[2] = 0;
                a[3] = 0;
            }
            else if (a[0] > 255)
            {
```

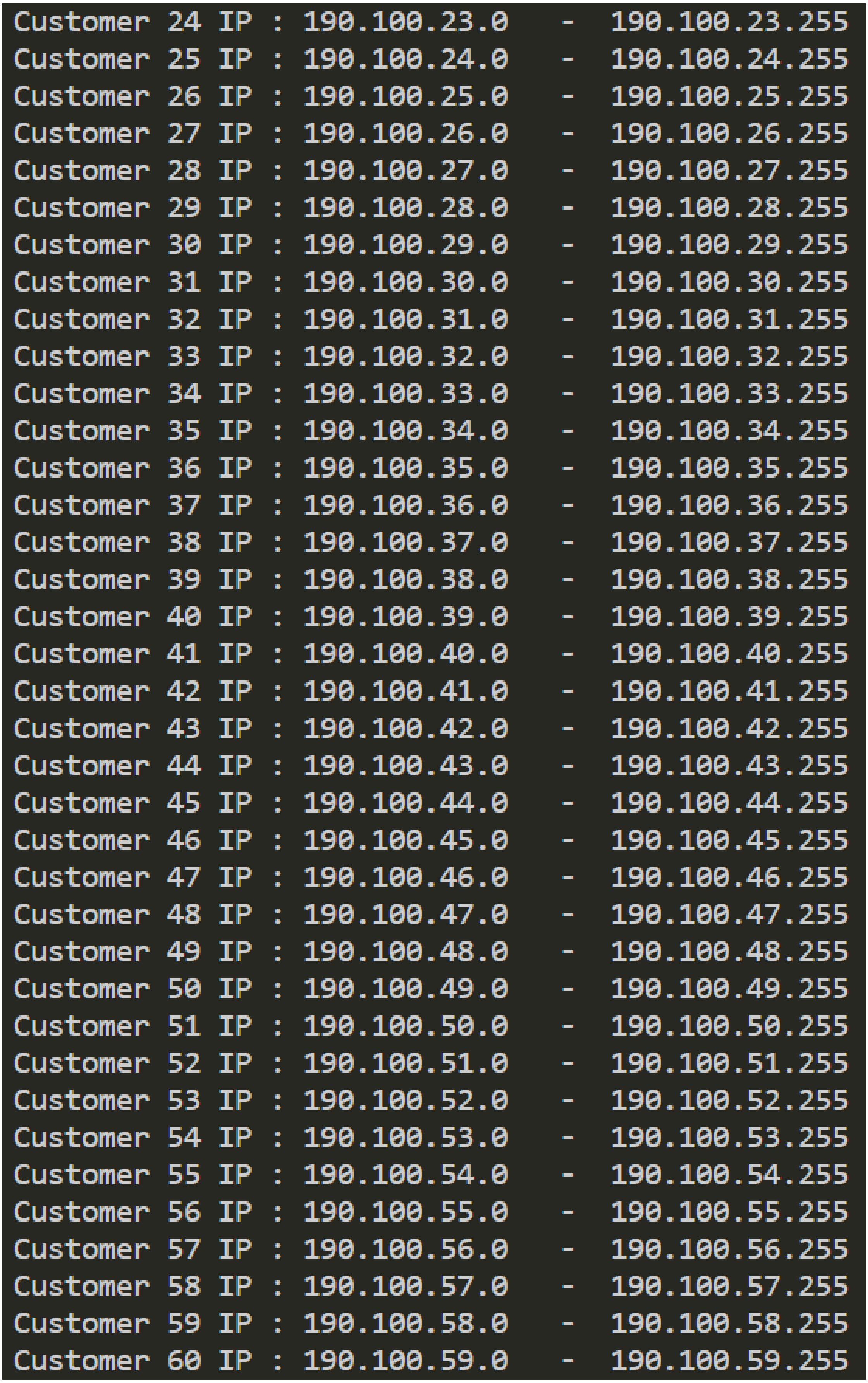


```
                printf("After this address cannot be allocated.");
                return 0;
            }
        }
    }
    cout << "Total Addres Allocated are: " << sum << endl;
    return 0;
}
```

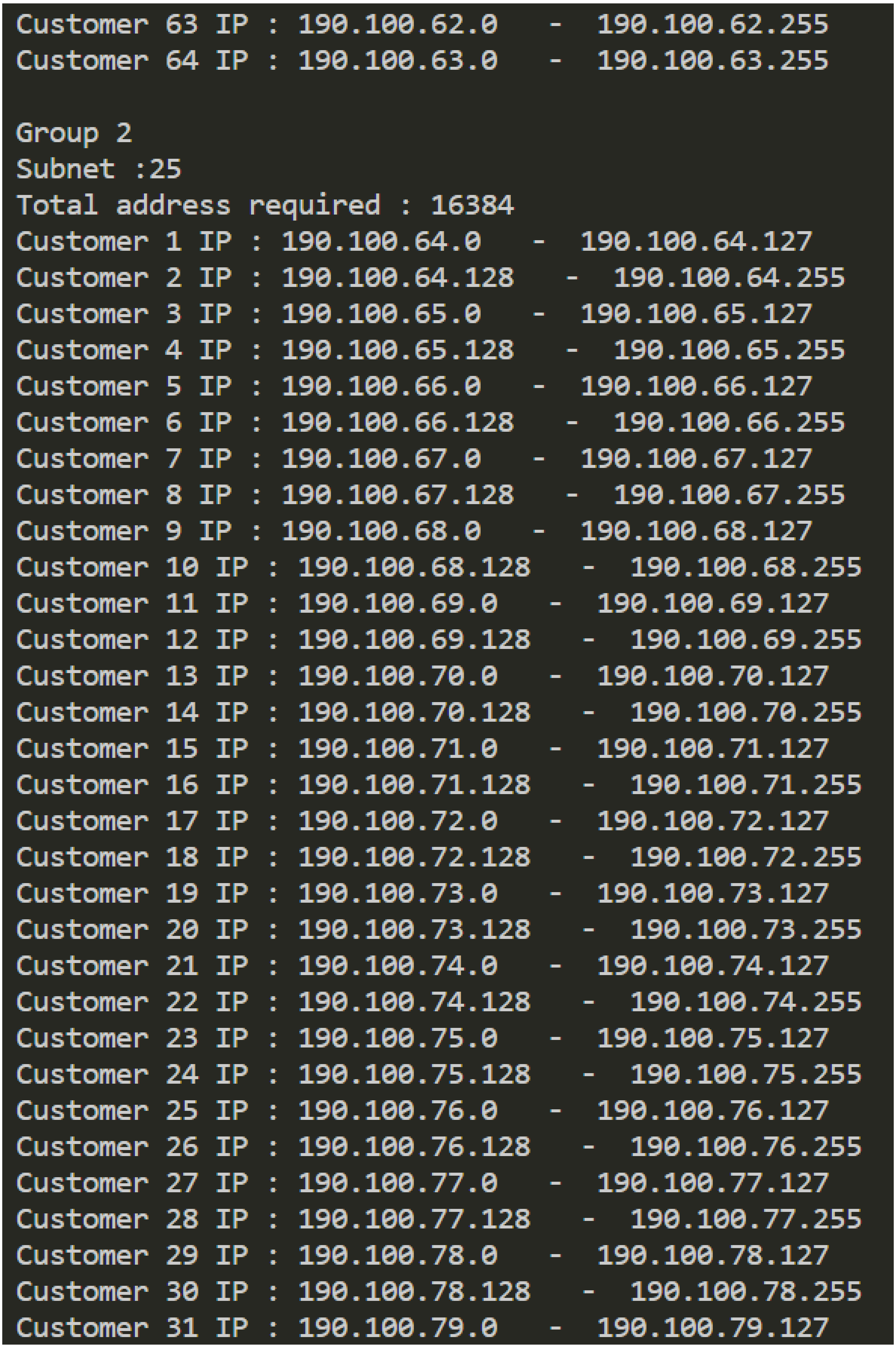
Input – Output:



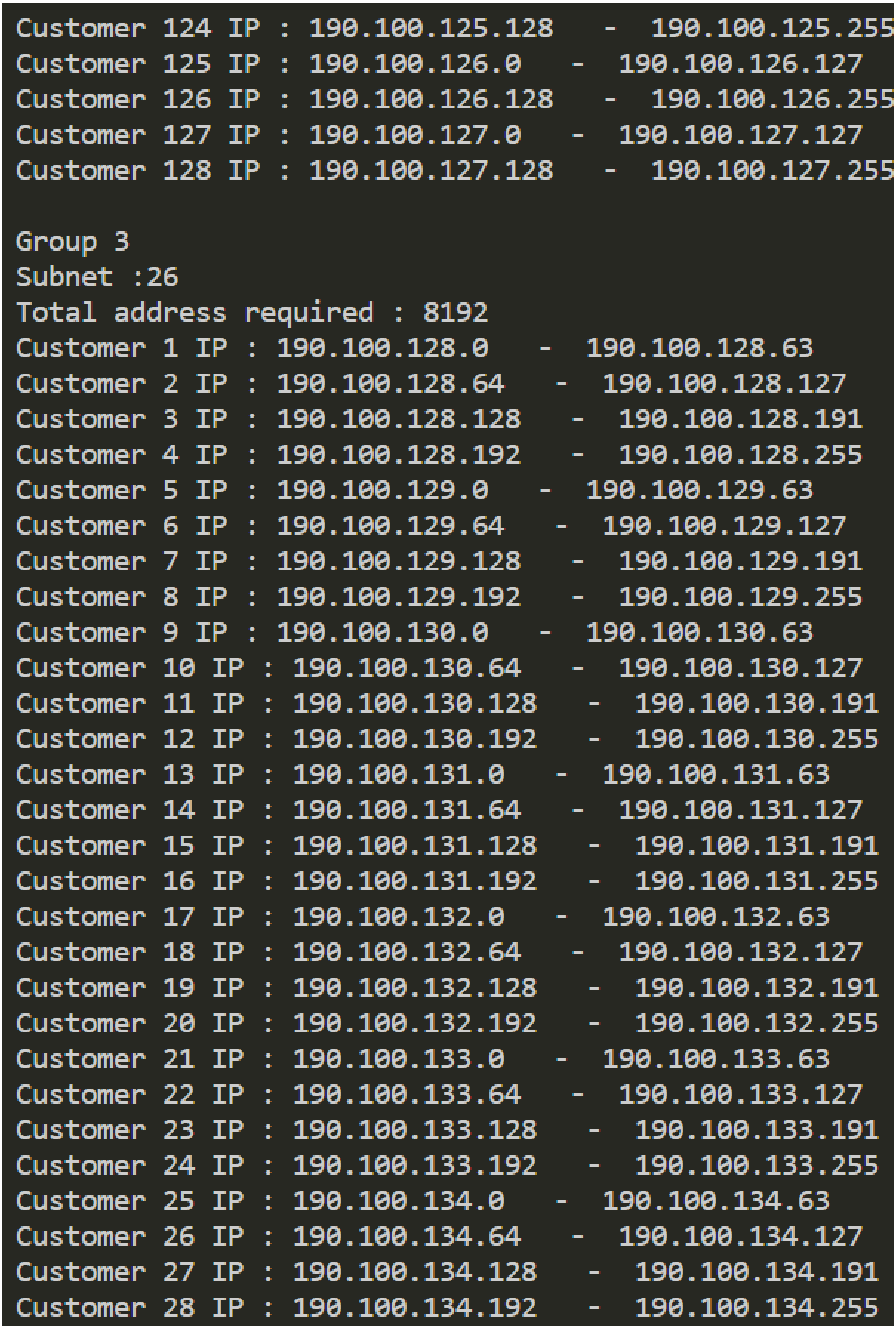
```
Enter the base address given :
 190.100.0.0/16
Total number of groups :
 3
Customers : 64
Need of addresses : 256
Customers : 128
Need of addresses : 128
Customers : 128
Need of addresses : 64

Group 1
Subnet :24
Total address required : 16384
Customer 1 IP : 190.100.0.0   -  190.100.0.255
Customer 2 IP : 190.100.1.0   -  190.100.1.255
Customer 3 IP : 190.100.2.0   -  190.100.2.255
Customer 4 IP : 190.100.3.0   -  190.100.3.255
Customer 5 IP : 190.100.4.0   -  190.100.4.255
Customer 6 IP : 190.100.5.0   -  190.100.5.255
Customer 7 IP : 190.100.6.0   -  190.100.6.255
Customer 8 IP : 190.100.7.0   -  190.100.7.255
Customer 9 IP : 190.100.8.0   -  190.100.8.255
Customer 10 IP : 190.100.9.0   -  190.100.9.255
Customer 11 IP : 190.100.10.0   -  190.100.10.255
Customer 12 IP : 190.100.11.0   -  190.100.11.255
Customer 13 IP : 190.100.12.0   -  190.100.12.255
Customer 14 IP : 190.100.13.0   -  190.100.13.255
Customer 15 IP : 190.100.14.0   -  190.100.14.255
Customer 16 IP : 190.100.15.0   -  190.100.15.255
Customer 17 IP : 190.100.16.0   -  190.100.16.255
Customer 18 IP : 190.100.17.0   -  190.100.17.255
Customer 19 IP : 190.100.18.0   -  190.100.18.255
Customer 20 IP : 190.100.19.0   -  190.100.19.255
Customer 21 IP : 190.100.20.0   -  190.100.20.255
Customer 22 IP : 190.100.21.0   -  190.100.21.255
Customer 23 IP : 190.100.22.0   -  190.100.22.255
Customer 24 IP : 190.100.23.0   -  190.100.23.255
```



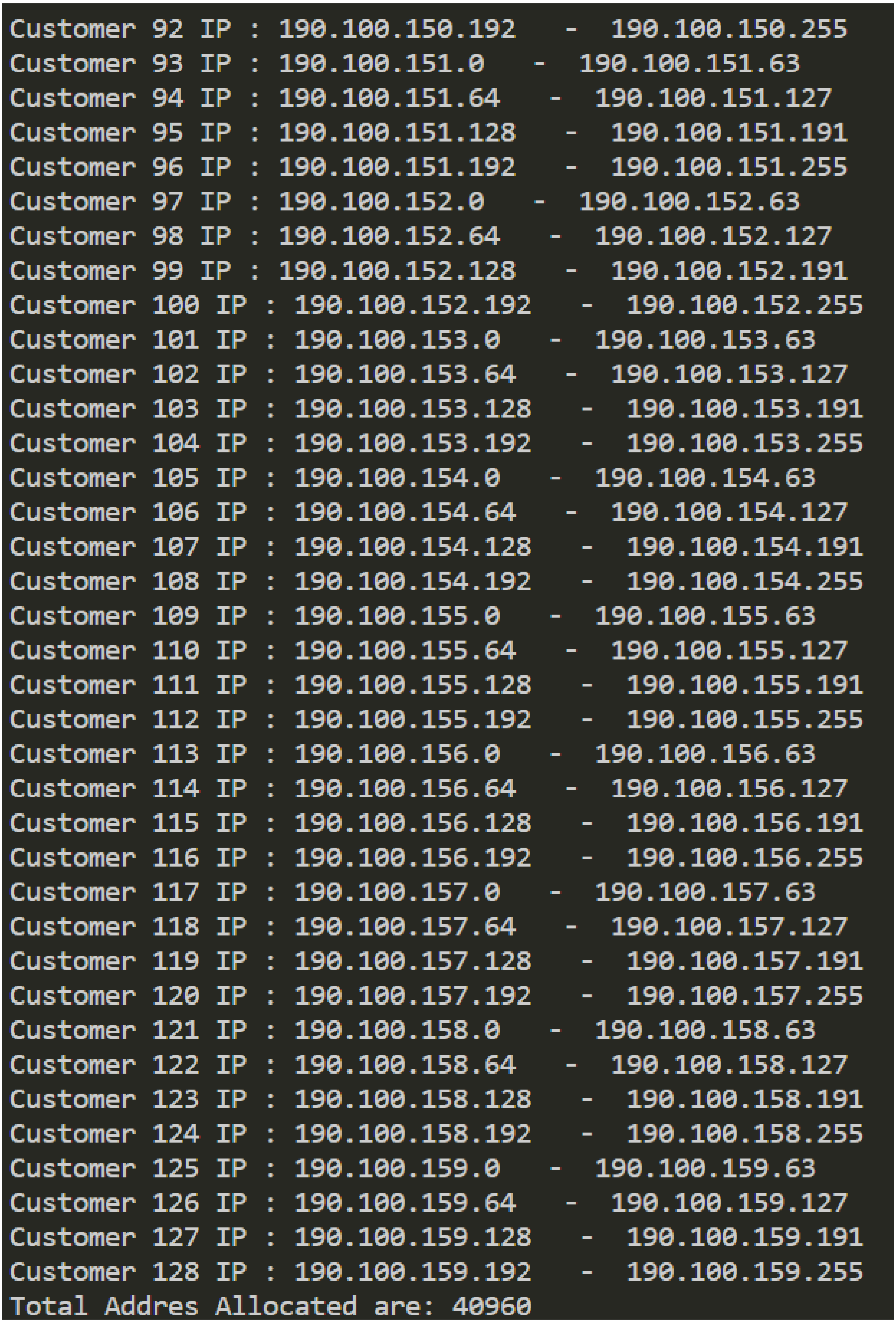
```
Customer 24 IP : 190.100.23.0   -   190.100.23.255
Customer 25 IP : 190.100.24.0   -   190.100.24.255
Customer 26 IP : 190.100.25.0   -   190.100.25.255
Customer 27 IP : 190.100.26.0   -   190.100.26.255
Customer 28 IP : 190.100.27.0   -   190.100.27.255
Customer 29 IP : 190.100.28.0   -   190.100.28.255
Customer 30 IP : 190.100.29.0   -   190.100.29.255
Customer 31 IP : 190.100.30.0   -   190.100.30.255
Customer 32 IP : 190.100.31.0   -   190.100.31.255
Customer 33 IP : 190.100.32.0   -   190.100.32.255
Customer 34 IP : 190.100.33.0   -   190.100.33.255
Customer 35 IP : 190.100.34.0   -   190.100.34.255
Customer 36 IP : 190.100.35.0   -   190.100.35.255
Customer 37 IP : 190.100.36.0   -   190.100.36.255
Customer 38 IP : 190.100.37.0   -   190.100.37.255
Customer 39 IP : 190.100.38.0   -   190.100.38.255
Customer 40 IP : 190.100.39.0   -   190.100.39.255
Customer 41 IP : 190.100.40.0   -   190.100.40.255
Customer 42 IP : 190.100.41.0   -   190.100.41.255
Customer 43 IP : 190.100.42.0   -   190.100.42.255
Customer 44 IP : 190.100.43.0   -   190.100.43.255
Customer 45 IP : 190.100.44.0   -   190.100.44.255
Customer 46 IP : 190.100.45.0   -   190.100.45.255
Customer 47 IP : 190.100.46.0   -   190.100.46.255
Customer 48 IP : 190.100.47.0   -   190.100.47.255
Customer 49 IP : 190.100.48.0   -   190.100.48.255
Customer 50 IP : 190.100.49.0   -   190.100.49.255
Customer 51 IP : 190.100.50.0   -   190.100.50.255
Customer 52 IP : 190.100.51.0   -   190.100.51.255
Customer 53 IP : 190.100.52.0   -   190.100.52.255
Customer 54 IP : 190.100.53.0   -   190.100.53.255
Customer 55 IP : 190.100.54.0   -   190.100.54.255
Customer 56 IP : 190.100.55.0   -   190.100.55.255
Customer 57 IP : 190.100.56.0   -   190.100.56.255
Customer 58 IP : 190.100.57.0   -   190.100.57.255
Customer 59 IP : 190.100.58.0   -   190.100.58.255
Customer 60 IP : 190.100.59.0   -   190.100.59.255
```



```
Customer 63 IP : 190.100.62.0   -  190.100.62.255
Customer 64 IP : 190.100.63.0   -  190.100.63.255

Group 2
Subnet :25
Total address required : 16384
Customer 1 IP : 190.100.64.0   -  190.100.64.127
Customer 2 IP : 190.100.64.128   -  190.100.64.255
Customer 3 IP : 190.100.65.0   -  190.100.65.127
Customer 4 IP : 190.100.65.128   -  190.100.65.255
Customer 5 IP : 190.100.66.0   -  190.100.66.127
Customer 6 IP : 190.100.66.128   -  190.100.66.255
Customer 7 IP : 190.100.67.0   -  190.100.67.127
Customer 8 IP : 190.100.67.128   -  190.100.67.255
Customer 9 IP : 190.100.68.0   -  190.100.68.127
Customer 10 IP : 190.100.68.128   -  190.100.68.255
Customer 11 IP : 190.100.69.0   -  190.100.69.127
Customer 12 IP : 190.100.69.128   -  190.100.69.255
Customer 13 IP : 190.100.70.0   -  190.100.70.127
Customer 14 IP : 190.100.70.128   -  190.100.70.255
Customer 15 IP : 190.100.71.0   -  190.100.71.127
Customer 16 IP : 190.100.71.128   -  190.100.71.255
Customer 17 IP : 190.100.72.0   -  190.100.72.127
Customer 18 IP : 190.100.72.128   -  190.100.72.255
Customer 19 IP : 190.100.73.0   -  190.100.73.127
Customer 20 IP : 190.100.73.128   -  190.100.73.255
Customer 21 IP : 190.100.74.0   -  190.100.74.127
Customer 22 IP : 190.100.74.128   -  190.100.74.255
Customer 23 IP : 190.100.75.0   -  190.100.75.127
Customer 24 IP : 190.100.75.128   -  190.100.75.255
Customer 25 IP : 190.100.76.0   -  190.100.76.127
Customer 26 IP : 190.100.76.128   -  190.100.76.255
Customer 27 IP : 190.100.77.0   -  190.100.77.127
Customer 28 IP : 190.100.77.128   -  190.100.77.255
Customer 29 IP : 190.100.78.0   -  190.100.78.127
Customer 30 IP : 190.100.78.128   -  190.100.78.255
Customer 31 IP : 190.100.79.0   -  190.100.79.127
```



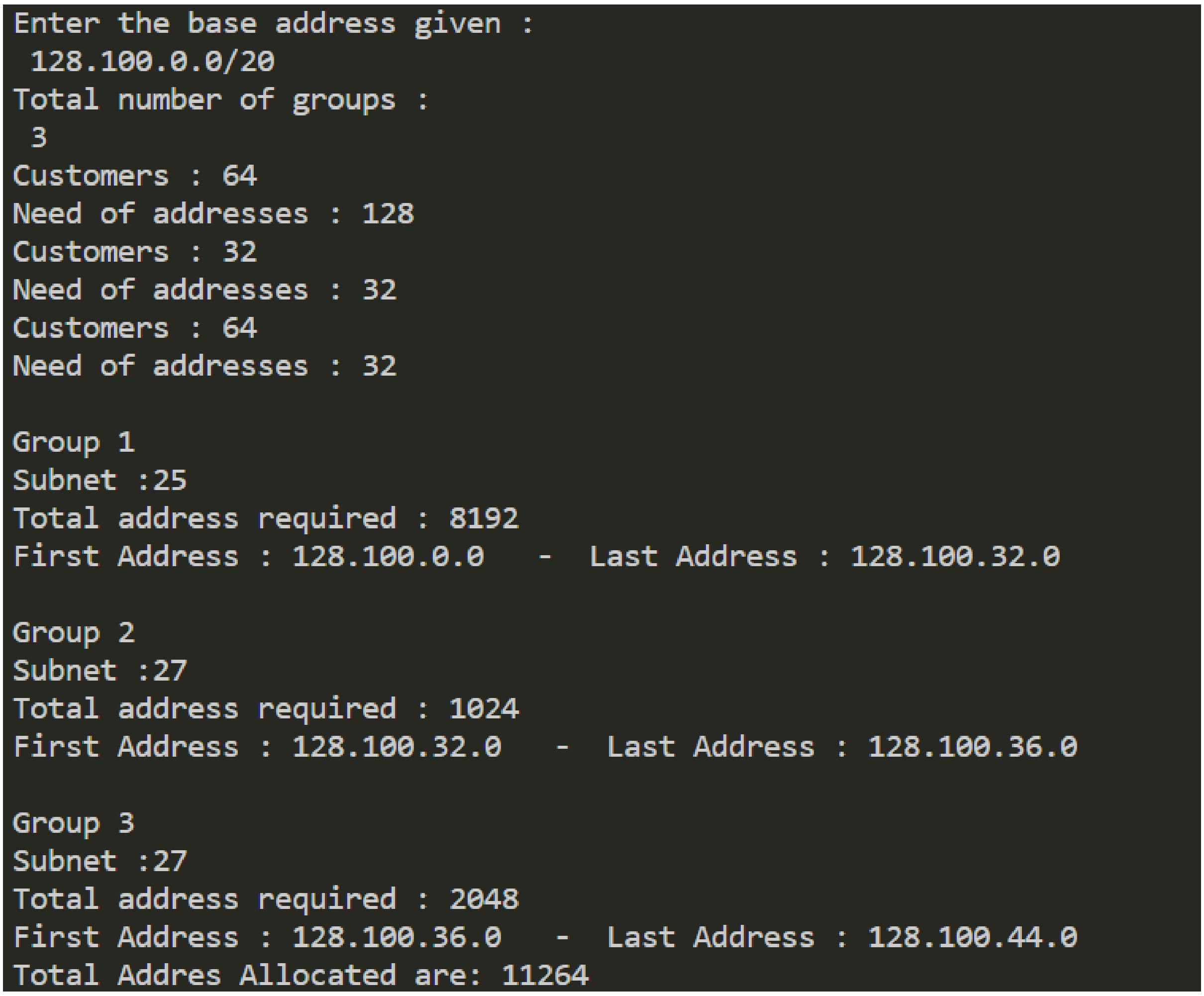
```
Customer 124 IP : 190.100.125.128   -  190.100.125.255
Customer 125 IP : 190.100.126.0   -  190.100.126.127
Customer 126 IP : 190.100.126.128   -  190.100.126.255
Customer 127 IP : 190.100.127.0   -  190.100.127.127
Customer 128 IP : 190.100.127.128   -  190.100.127.255

Group 3
Subnet :26
Total address required : 8192
Customer 1 IP : 190.100.128.0   -  190.100.128.63
Customer 2 IP : 190.100.128.64   -  190.100.128.127
Customer 3 IP : 190.100.128.128   -  190.100.128.191
Customer 4 IP : 190.100.128.192   -  190.100.128.255
Customer 5 IP : 190.100.129.0   -  190.100.129.63
Customer 6 IP : 190.100.129.64   -  190.100.129.127
Customer 7 IP : 190.100.129.128   -  190.100.129.191
Customer 8 IP : 190.100.129.192   -  190.100.129.255
Customer 9 IP : 190.100.130.0   -  190.100.130.63
Customer 10 IP : 190.100.130.64   -  190.100.130.127
Customer 11 IP : 190.100.130.128   -  190.100.130.191
Customer 12 IP : 190.100.130.192   -  190.100.130.255
Customer 13 IP : 190.100.131.0   -  190.100.131.63
Customer 14 IP : 190.100.131.64   -  190.100.131.127
Customer 15 IP : 190.100.131.128   -  190.100.131.191
Customer 16 IP : 190.100.131.192   -  190.100.131.255
Customer 17 IP : 190.100.132.0   -  190.100.132.63
Customer 18 IP : 190.100.132.64   -  190.100.132.127
Customer 19 IP : 190.100.132.128   -  190.100.132.191
Customer 20 IP : 190.100.132.192   -  190.100.132.255
Customer 21 IP : 190.100.133.0   -  190.100.133.63
Customer 22 IP : 190.100.133.64   -  190.100.133.127
Customer 23 IP : 190.100.133.128   -  190.100.133.191
Customer 24 IP : 190.100.133.192   -  190.100.133.255
Customer 25 IP : 190.100.134.0   -  190.100.134.63
Customer 26 IP : 190.100.134.64   -  190.100.134.127
Customer 27 IP : 190.100.134.128   -  190.100.134.191
Customer 28 IP : 190.100.134.192   -  190.100.134.255
```



```
Customer 92 IP : 190.100.150.192   -  190.100.150.255
Customer 93 IP : 190.100.151.0   -  190.100.151.63
Customer 94 IP : 190.100.151.64   -  190.100.151.127
Customer 95 IP : 190.100.151.128   -  190.100.151.191
Customer 96 IP : 190.100.151.192   -  190.100.151.255
Customer 97 IP : 190.100.152.0   -  190.100.152.63
Customer 98 IP : 190.100.152.64   -  190.100.152.127
Customer 99 IP : 190.100.152.128   -  190.100.152.191
Customer 100 IP : 190.100.152.192   -  190.100.152.255
Customer 101 IP : 190.100.153.0   -  190.100.153.63
Customer 102 IP : 190.100.153.64   -  190.100.153.127
Customer 103 IP : 190.100.153.128   -  190.100.153.191
Customer 104 IP : 190.100.153.192   -  190.100.153.255
Customer 105 IP : 190.100.154.0   -  190.100.154.63
Customer 106 IP : 190.100.154.64   -  190.100.154.127
Customer 107 IP : 190.100.154.128   -  190.100.154.191
Customer 108 IP : 190.100.154.192   -  190.100.154.255
Customer 109 IP : 190.100.155.0   -  190.100.155.63
Customer 110 IP : 190.100.155.64   -  190.100.155.127
Customer 111 IP : 190.100.155.128   -  190.100.155.191
Customer 112 IP : 190.100.155.192   -  190.100.155.255
Customer 113 IP : 190.100.156.0   -  190.100.156.63
Customer 114 IP : 190.100.156.64   -  190.100.156.127
Customer 115 IP : 190.100.156.128   -  190.100.156.191
Customer 116 IP : 190.100.156.192   -  190.100.156.255
Customer 117 IP : 190.100.157.0   -  190.100.157.63
Customer 118 IP : 190.100.157.64   -  190.100.157.127
Customer 119 IP : 190.100.157.128   -  190.100.157.191
Customer 120 IP : 190.100.157.192   -  190.100.157.255
Customer 121 IP : 190.100.158.0   -  190.100.158.63
Customer 122 IP : 190.100.158.64   -  190.100.158.127
Customer 123 IP : 190.100.158.128   -  190.100.158.191
Customer 124 IP : 190.100.158.192   -  190.100.158.255
Customer 125 IP : 190.100.159.0   -  190.100.159.63
Customer 126 IP : 190.100.159.64   -  190.100.159.127
Customer 127 IP : 190.100.159.128   -  190.100.159.191
Customer 128 IP : 190.100.159.192   -  190.100.159.255
Total Addres Allocated are: 40960
```

Vandit Gabani
20BCI0090

Ex – 2



```
Enter the base address given :
 128.100.0.0/20
Total number of groups :
 3
Customers : 64
Need of addresses : 128
Customers : 32
Need of addresses : 32
Customers : 64
Need of addresses : 32

Group 1
Subnet :25
Total address required : 8192
First Address : 128.100.0.0   -  Last Address : 128.100.32.0

Group 2
Subnet :27
Total address required : 1024
First Address : 128.100.32.0   -  Last Address : 128.100.36.0

Group 3
Subnet :27
Total address required : 2048
First Address : 128.100.36.0   -  Last Address : 128.100.44.0
Total Addres Allocated are: 11264
```