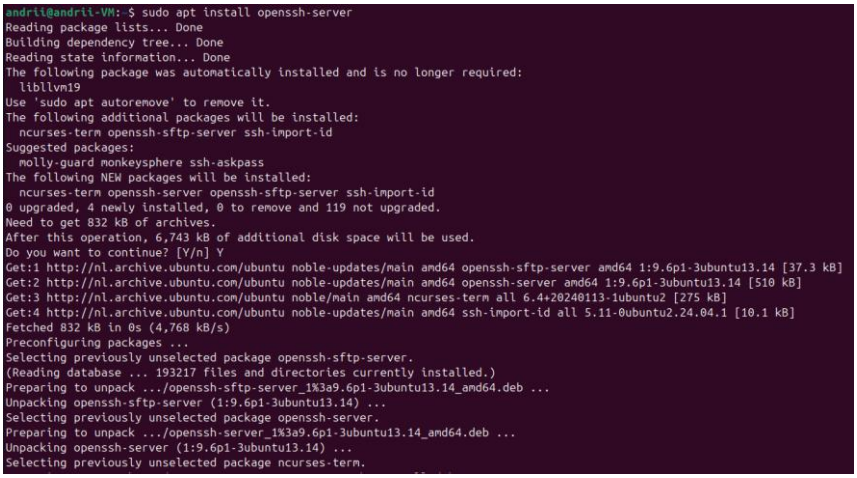
Week 6 – Networking

Student number: 578856

Assignment 6.1: Working from home

Screenshot installation openssh-server:



Screenshot successful SSH command execution:

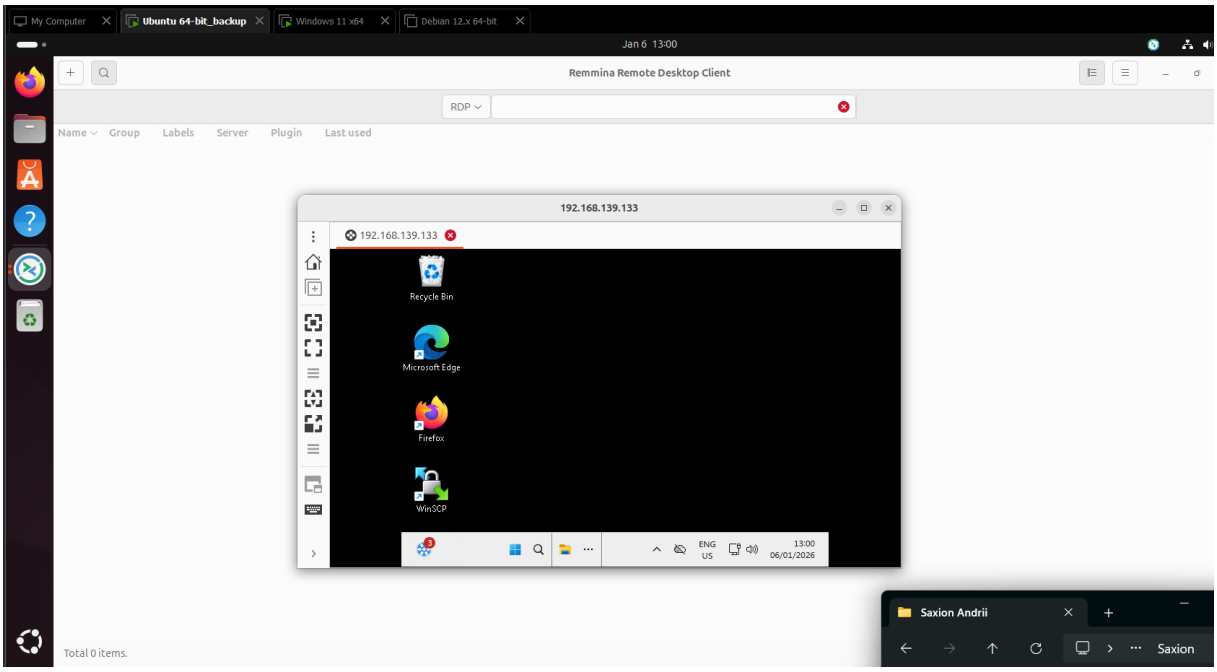
```
C:\Users\user>ssh andrii@192.168.139.136
The authenticity of host '192.168.139.136 (192.168.139.136)' can't be established.
ED25519 key fingerprint is SHA256:nPsGz1wThqpMy4dsiVXSEKlPQl9njX8A5dNx1uHedj4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added '192.168.139.136' (ED25519) to the list of known hosts.
andrii@192.168.139.136's password:
Permission denied, please try again.
andrii@192.168.139.136's password:
Permission denied, please try again.
andrii@192.168.139.136's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

59 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

17 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

andrii@andrii-VM:~$
```

Screenshot successful execution SCP command:

```
andrii@andrii-VM:~$ exit
logout
Connection to 192.168.139.136 closed.

C:\Users\user>scp C:\Users\aleho\test.txt andrii@192.168.139.136 C:/home/andrii/
C:/home/andrii/: No such file or directory

C:\Users\user>scp C:\Users\aleho\test.txt andrii@192.168.139.136:/home/andrii/
scp: stat local "C:/Users/aleho/test.txt": No such file or directory

C:\Users\user>scp C:\Users\user\test.txt andrii@192.168.139.136:/home/andrii/
andrii@192.168.139.136's password:
test.txt

C:\Users\user>
```

```
andrii@andrii-VM:~$ ls
archiveFolder  Desktop    Downloads         file.txt  Music    output.gif  ppp.txt  sherlock.txt  Templates  Videos
archive.tar    Documents  email-base64.txt  hello     oldcar.  Pictures    Public   snap          test.txt
andrii@andrii-VM:~$
```

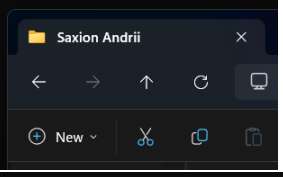
Screenshot remmina:



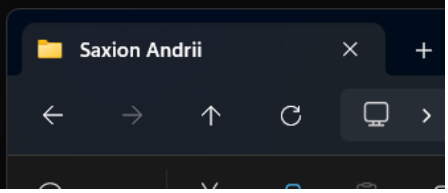
Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:

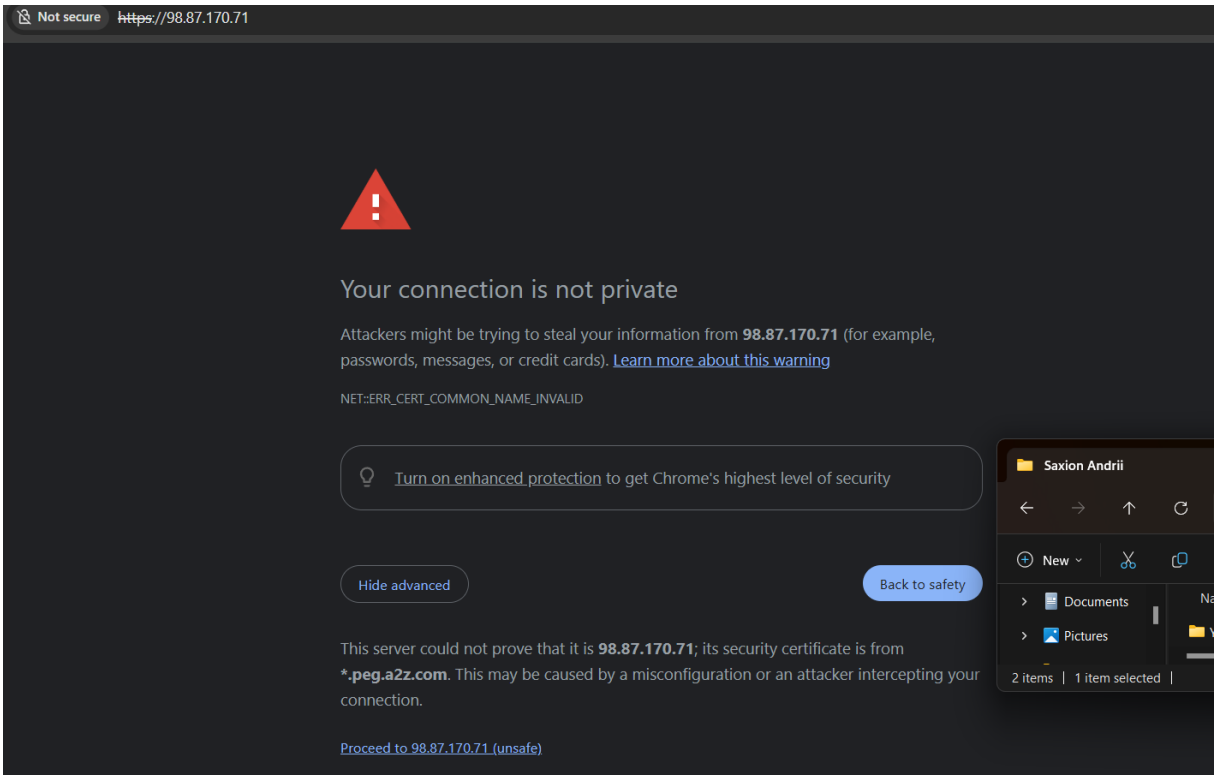
```
Microsoft Windows [Version 10.0.26100.7462]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aleho>nslookup
Default Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

> amazon.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    amazon.com
Addresses:  98.87.170.71
          98.82.161.185
          98.87.170.74

> google.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    google.com
Addresses:  2a00:1450:400e:804::200e
          142.251.39.142

> one.one.one.one
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    one.one.one.one
Addresses:  2606:4700:4700::1111
          2606:4700:4700::1001
          1.0.0.1
          1.1.1.1

> dns.google.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10
```




```
> dns.google.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    dns.google.com
Addresses:  2001:4860:4860::8888
          2001:4860:4860::8844
          8.8.8.8
          8.8.4.4

> bol.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    bol.com
Address:  79.170.100.62

> w3schools.com
Server:  d-hk-mer-ib02.infra.saxion.net
Address:  145.2.14.10

Non-authoritative answer:
Name:    w3schools.com
Addresses:  76.223.115.82
          13.248.240.135

> exit

C:\Users\aleho>
```



Screenshot website visit via IP address:



Assignment 6.3: subnetting

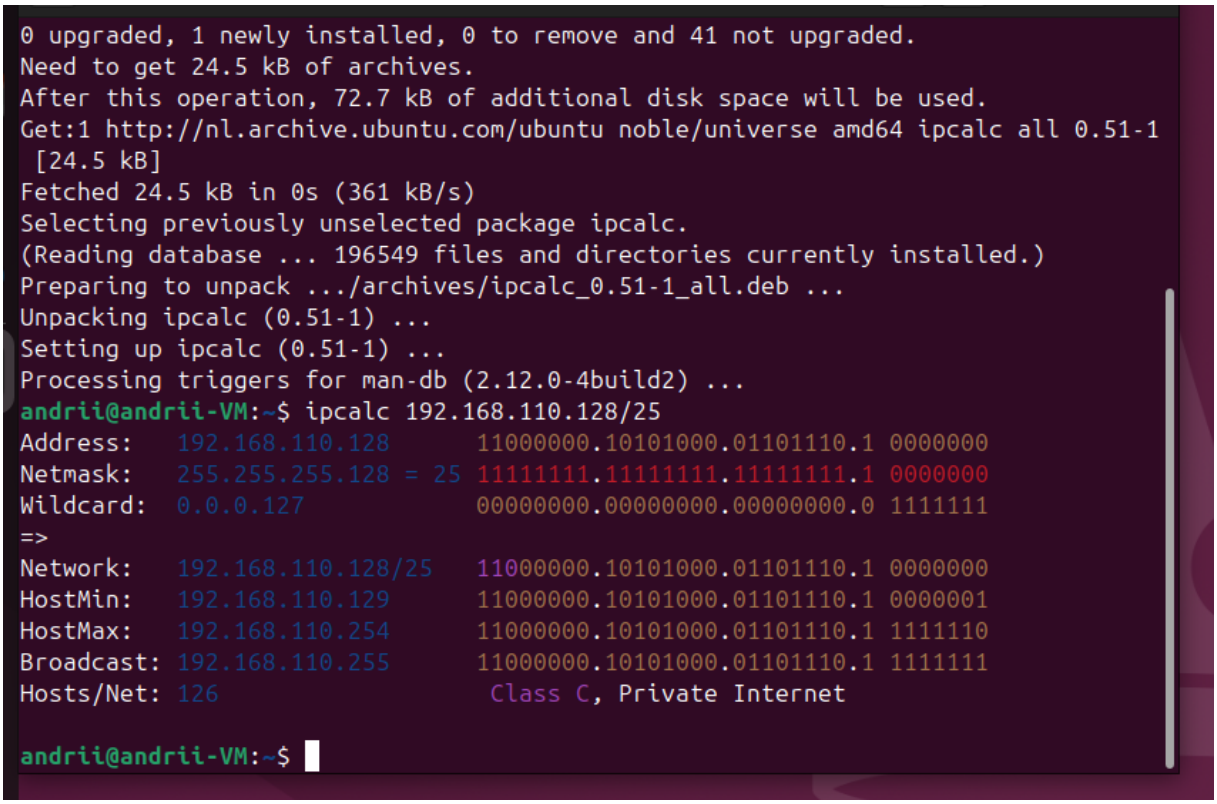
How many IP addresses are in this network configuration 192.168.110.128/25?

There are 128 IP addresses in total, because 7 bits are used for host addresses and 2 to the power of 7 is 128.

What is the usable IP range to hand out to the connected computers?

The usable IP range is from 192.168.110.129 to 192.168.110.254, so 126 usable IPs.

Check your two previous answers with this Linux command: `ipcalc 192.168.110.128/25`



Explain the above calculation in your own words.

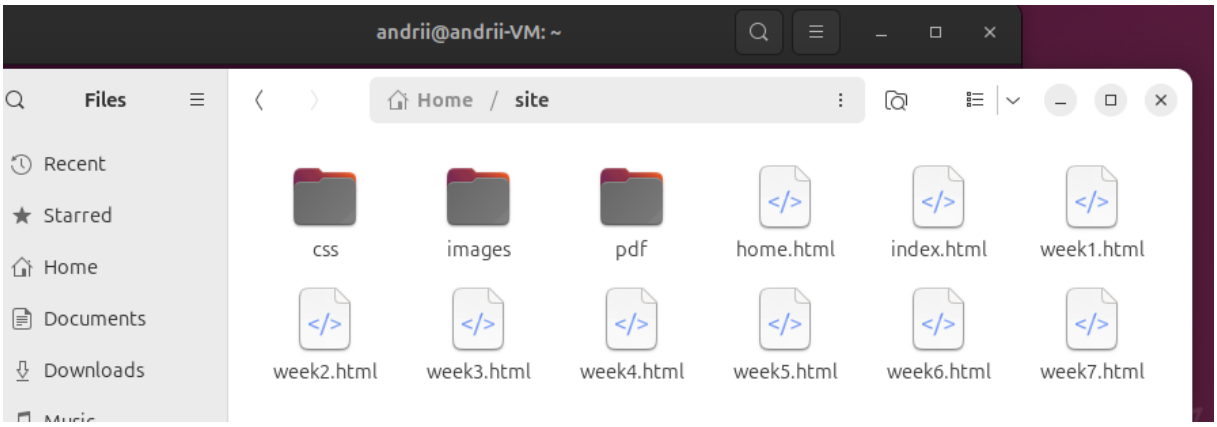
The subnet 192.168.110.128/25 starts at .128 and ends at .255. The first address is the network address and the last one is the broadcast address, so they cannot be used by computers. That leaves 126 usable IP addresses that can be assigned to devices in the network. The /25 means that first 25 bits are used to identify the network and last 7 is to identify the device.

Assignment 6.4: HTML

Screenshot IP address Ubuntu VM:

```
andrii@andrii-VM:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:85:4f:64 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.139.136/24 brd 192.168.139.255 scope global dynamic noprefixrou
te ens33
       valid_lft 915sec preferred_lft 915sec
    inet6 fe80::20c:29ff:fe85:4f64/64 scope link
       valid_lft forever preferred_lft forever
andrii@andrii-VM:~$
```

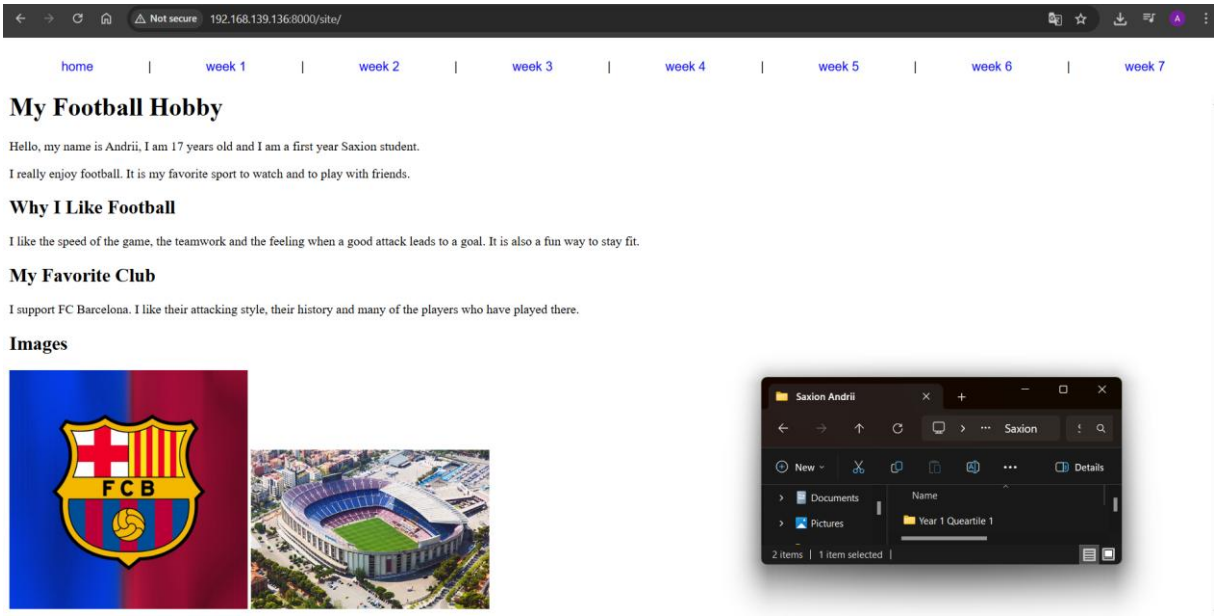
Screenshot of Site directory contents:



Screenshot python3 webserver command:

```
andrii@andrii-VM:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:85:4f:64 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.139.136/24 brd 192.168.139.255 scope global dynamic noprefixrou
te ens33
       valid_lft 915sec preferred_lft 915sec
    inet6 fe80::20c:29ff:fe85:4f64/64 scope link
       valid_lft forever preferred_lft forever
andrii@andrii-VM:~$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Screenshot web browser visits your site



Assignment 6.5: Network segment

Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-----------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```

Paste source code here, with a screenshot of a working application.

```
public class Main {
   public static void main(String[] args) {
      exercise1();
      exercise2();
      exerceise3();
      exerceise4();
      exerceise5();
      exerceise6();
      exerceise7();
      System.out.println(calculateNetworkAddress("192.168.1.100", "255.255.255.224"));
   }

   public static void  exercise1(){
      int number = 5;
      if((number & 1) == 1){
         System.out.println("number is odd");
      }else{
         System.out.println("number is even");
      }
   }
   public static void  exercise2(){
      int number = 3;
      if((number & (number-1)) == 0){
         System.out.println("number is a power of 2");
      }
      else System.out.println("number isn't a power of 2");
   }
   public static void exerceise3(){
      final int READ = 4;
      final int WRITE = 2;
      final int EXECUTE = 1;

      int userPermissions = 7;
      if ((userPermissions & READ) == 4 ){
```

```
        System.out.println("User has read permissions");
    }else {
        System.out.println("User can't read. No permissions.");
    }
}
public static void exerceise4(){
    final int READ = 4;
    final int WRITE = 2;
    final int EXECUTE = 1;

    int userPermissions = READ | EXECUTE;
    System.out.println("User permissions: "+userPermissions);
}

public static void exerceise5(){
    final int READ = 4;
    final int WRITE = 2;
    final int EXECUTE = 1;

    int userPermissions = 6;
    userPermissions = userPermissions ^ WRITE;
    System.out.println("User permissions: "+userPermissions);
}
public static void exerceise6(){
    int number = 5;
    number = ~ (number - 1);
    System.out.println("Number: " + number);
}
public static void exerceise7(){
    int number = 10;
    System.out.println("Decimal number is: " + number);

    String binary = Integer.toBinaryString(number);
    String octal = Integer.toOctalString(number);
    String hex = Integer.toHexString(number);

    System.out.println("Binary number: " + binary);
    System.out.println("Octal number: " + octal);
    System.out.println("Hex number: " + hex);
}
public static String calculateNetworkAddress(String ip, String subnet) {
    String[] ipParts = ip.split("\\.");
    String[] subnetParts = subnet.split("\\.");
    String network = "";

    for (int i = 0; i < ipParts.length; i++){
        int andOperation = Integer.parseInt(ipParts[i])  & Integer.parseInt(subnetParts[i]);
```

```
            if (i == ipParts.length - 1){
                network += andOperation;
            }else {
                network += andOperation + ".";
            }
        }

        return network;
    }
}
```