Template Week 6 – Networking

Student number:

571291

Assignment 6.1: Working from home

Screenshot installation openssh-server:

Screenshot successful SSH command execution:

Screenshot successful execution SCP command:

Screenshot remmina:

Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:

Screenshot website visit via IP address:

Assignment 6.3: subnetting

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

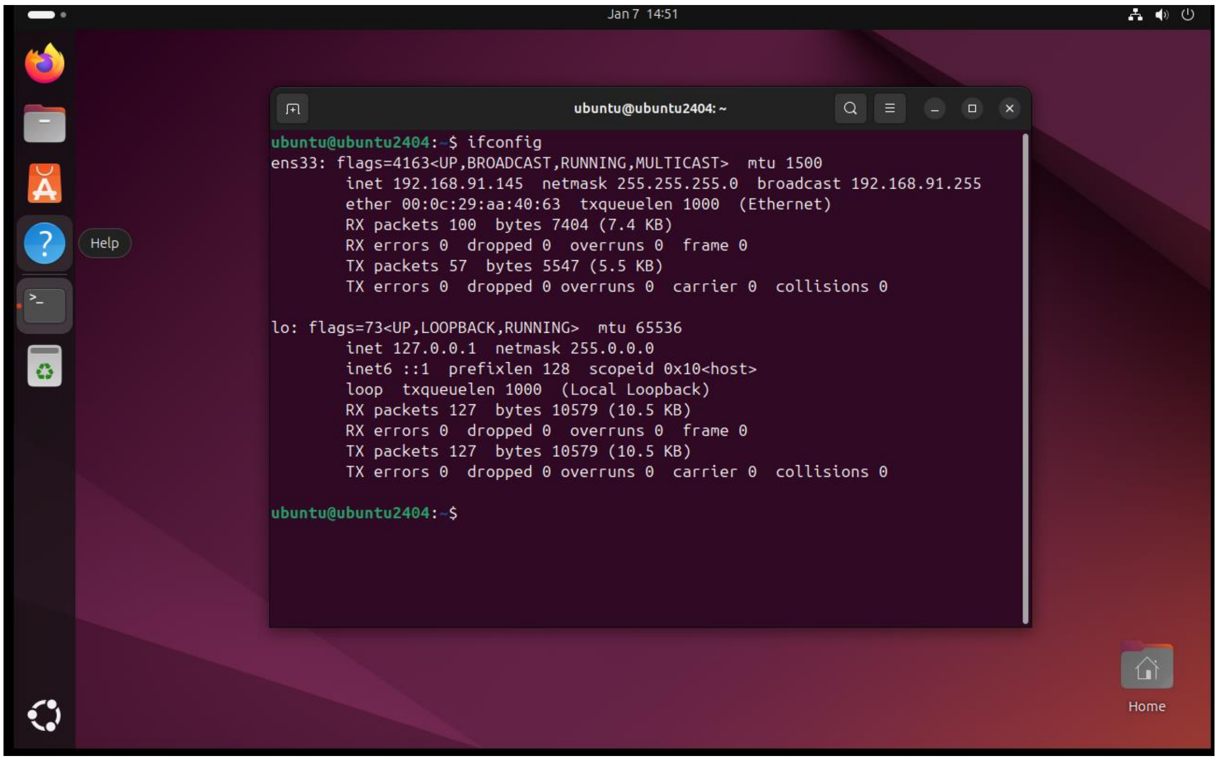
What is the usable IP range to hand out to the connected computers?

Check your two previous answers with this calculator:
https://www.calculator.net/ip-subnet-calculator.html

Explain the above calculation in your own words.

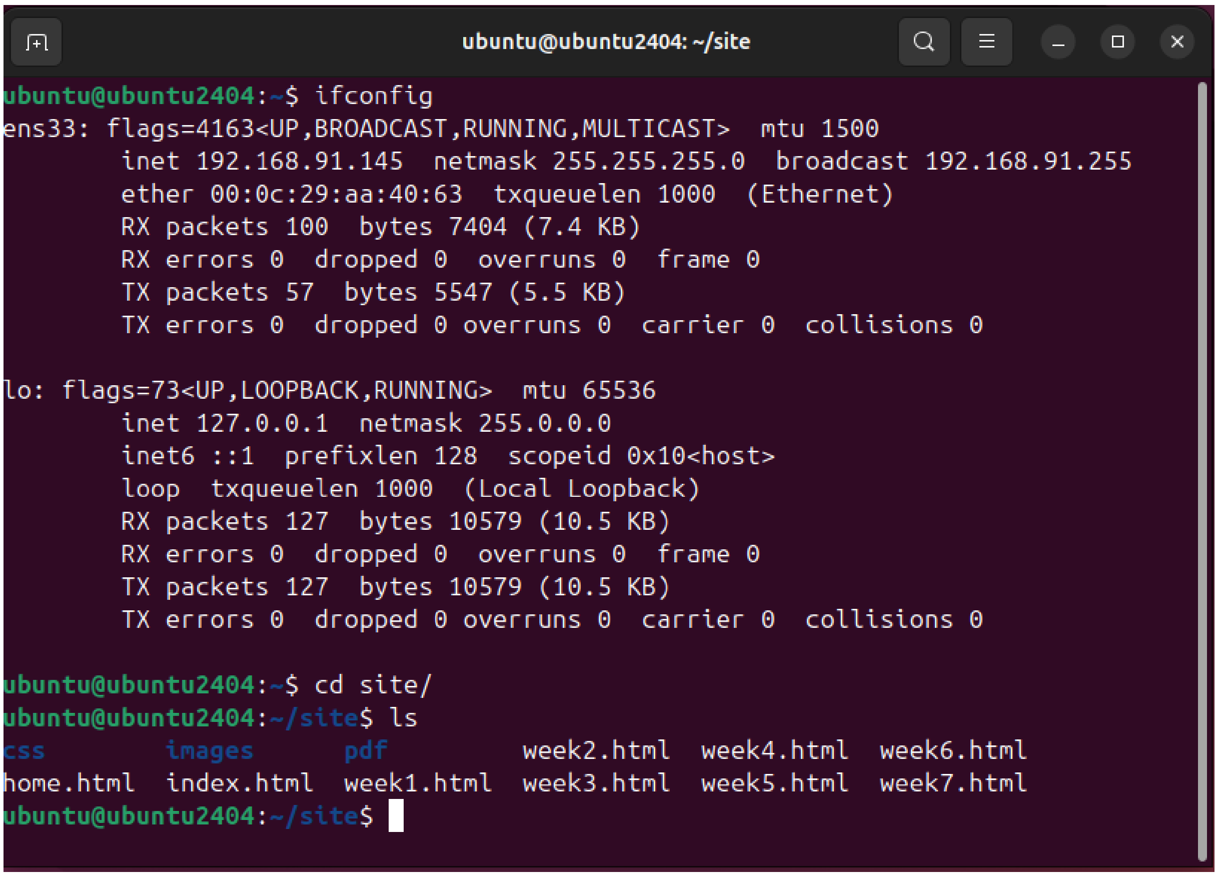
Assignment 6.4: HTML

Screenshot IP address Ubuntu VM:

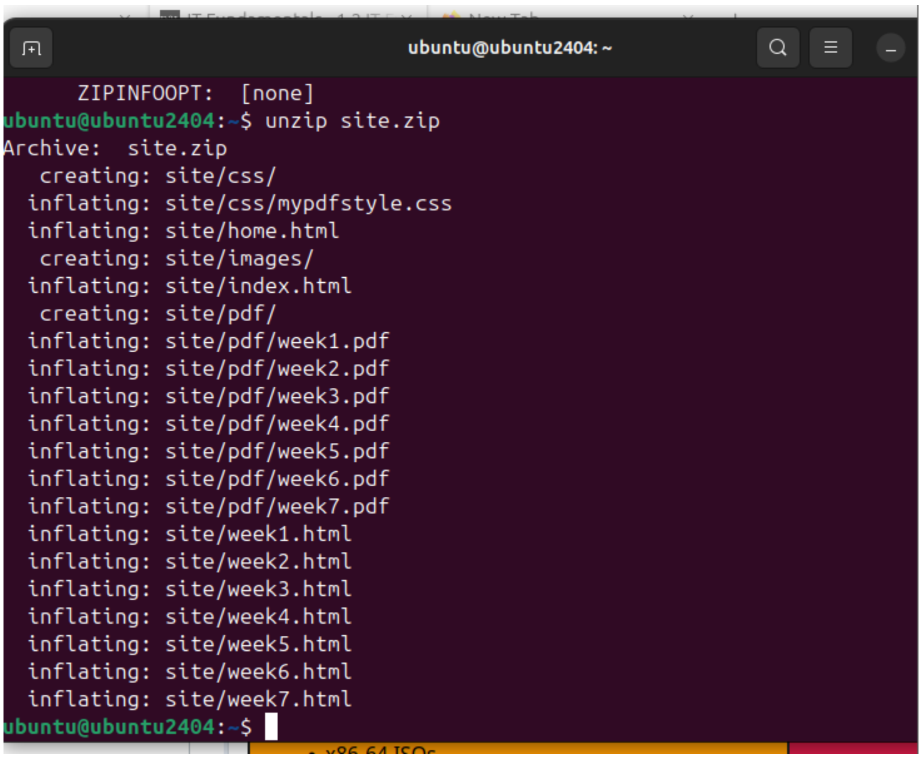


```
ubuntu@ubuntu2404:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.91.145  netmask 255.255.255.0  broadcast 192.168.91.255
        ether 00:0c:29:aa:40:63  txqueuelen 1000  (Ethernet)
        RX packets 100  bytes 7404 (7.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 5547 (5.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 127  bytes 10579 (10.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 127  bytes 10579 (10.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@ubuntu2404:~$
```

Screenshot of Site directory contents:

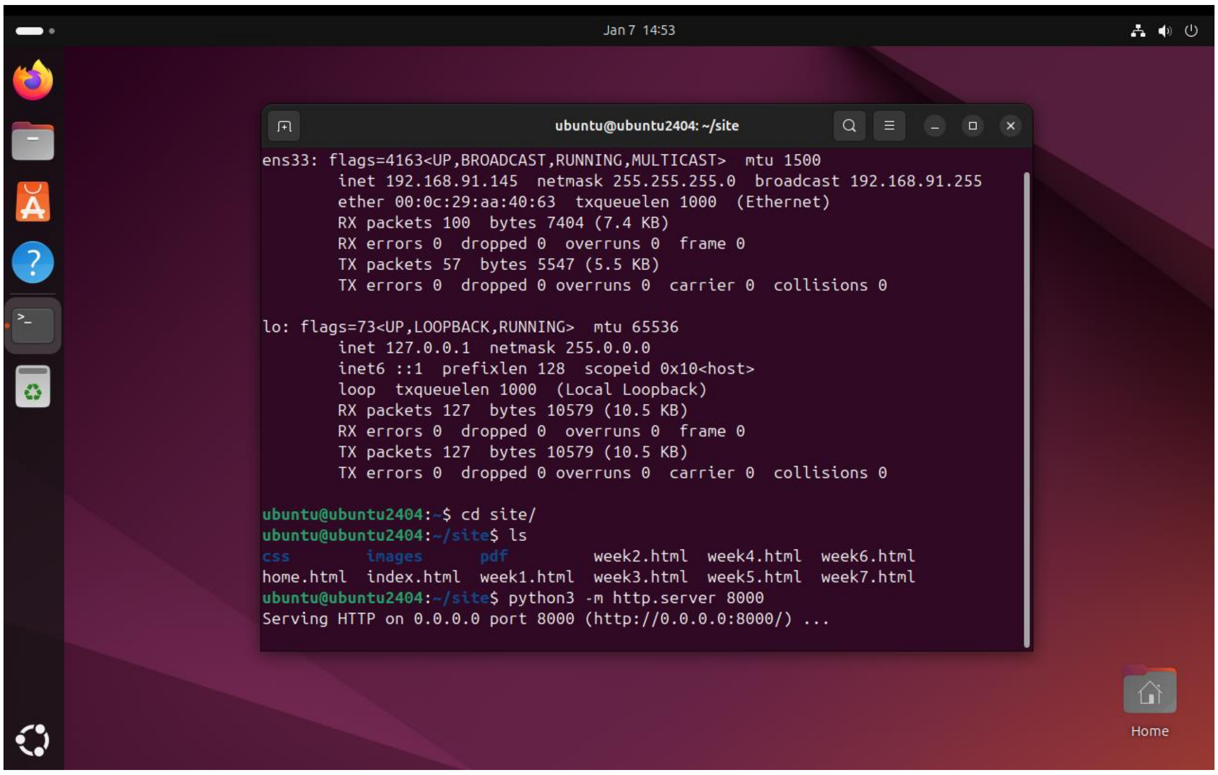


```
ubuntu@ubuntu2404:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.91.145  netmask 255.255.255.0  broadcast 192.168.91.255
        ether 00:0c:29:aa:40:63  txqueuelen 1000  (Ethernet)
        RX packets 100  bytes 7404 (7.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 5547 (5.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 127  bytes 10579 (10.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 127  bytes 10579 (10.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@ubuntu2404:~$ cd site/
ubuntu@ubuntu2404:~/site$ ls
css        images     pdf        week2.html  week4.html  week6.html
home.html  index.html  week1.html  week3.html  week5.html  week7.html
ubuntu@ubuntu2404:~/site$
```



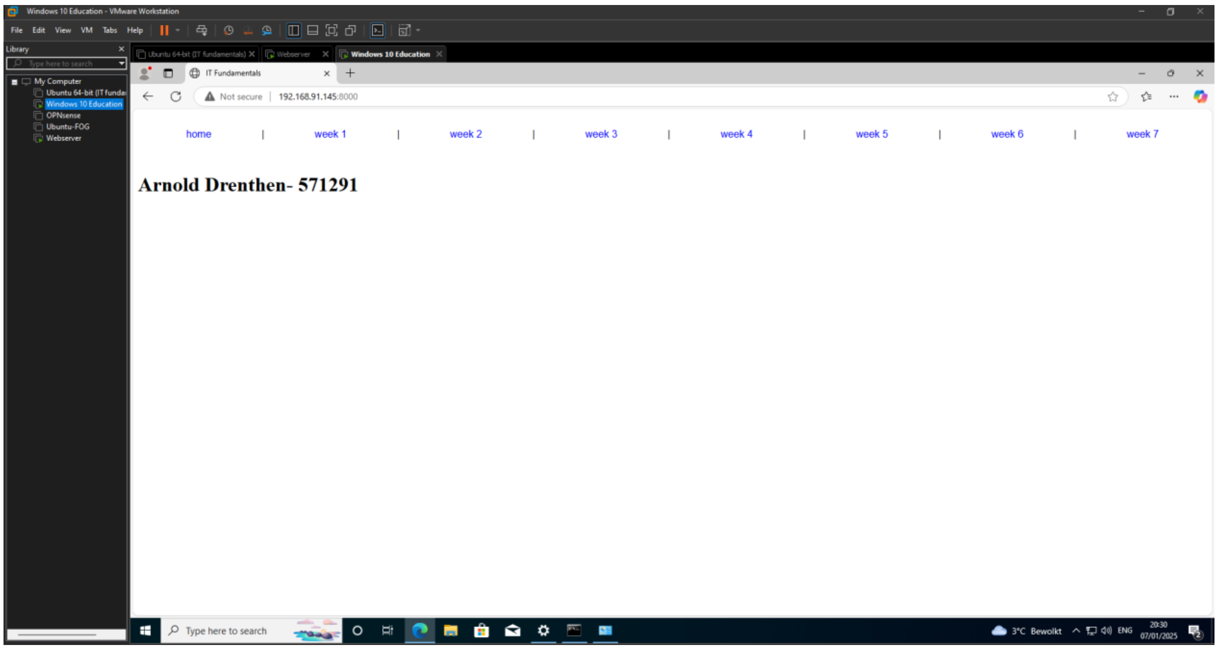
```
     ZIPINFOOPT:  [none]
ubuntu@ubuntu2404:~$ unzip site.zip
Archive:  site.zip
   creating: site/css/
  inflating: site/css/mypdfstyle.css
  inflating: site/home.html
   creating: site/images/
  inflating: site/index.html
   creating: site/pdf/
  inflating: site/pdf/week1.pdf
  inflating: site/pdf/week2.pdf
  inflating: site/pdf/week3.pdf
  inflating: site/pdf/week4.pdf
  inflating: site/pdf/week5.pdf
  inflating: site/pdf/week6.pdf
  inflating: site/pdf/week7.pdf
  inflating: site/week1.html
  inflating: site/week2.html
  inflating: site/week3.html
  inflating: site/week4.html
  inflating: site/week5.html
  inflating: site/week6.html
  inflating: site/week7.html
ubuntu@ubuntu2404:~$
```

Screenshot python3 webserver command:



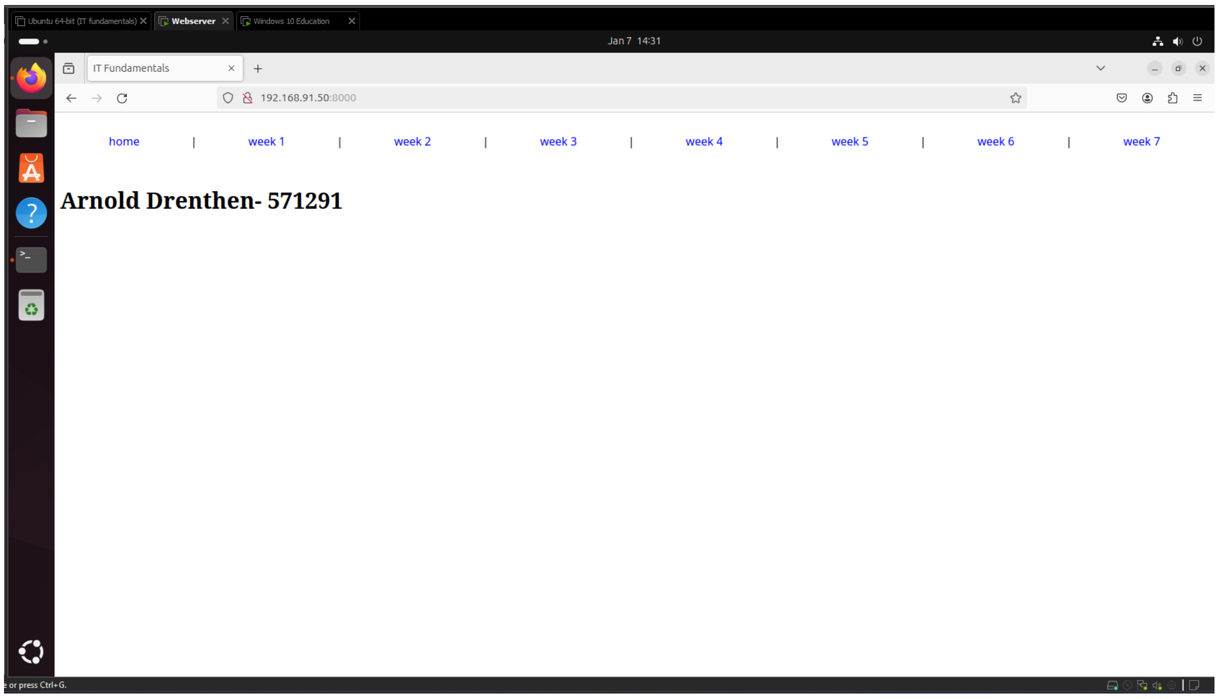
```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.91.145  netmask 255.255.255.0  broadcast 192.168.91.255
        ether 00:0c:29:aa:40:63  txqueuelen 1000  (Ethernet)
        RX packets 100  bytes 7404 (7.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 57  bytes 5547 (5.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 127  bytes 10579 (10.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 127  bytes 10579 (10.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ubuntu@ubuntu2404:~$ cd site/
ubuntu@ubuntu2404:~/site$ ls
css         images      pdf         week2.html  week4.html  week6.html
home.html   index.html  week1.html  week3.html  week5.html  week7.html
ubuntu@ubuntu2404:~/site$ 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



home | week 1 | week 2 | week 3 | week 4 | week 5 | week 6 | week 7

Arnold Drenthen- 571291


Ubuntu 64-bit (IT fundamentals) | Webserver | Windows 10 Education

Jan 7 14:31

IT Fundamentals

192.168.91.50:8000

home | week 1 | week 2 | week 3 | week 4 | week 5 | week 6 | week 7

Arnold Drenthen- 571291

or press Ctrl+G.

Bonus point assignment – week 6

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

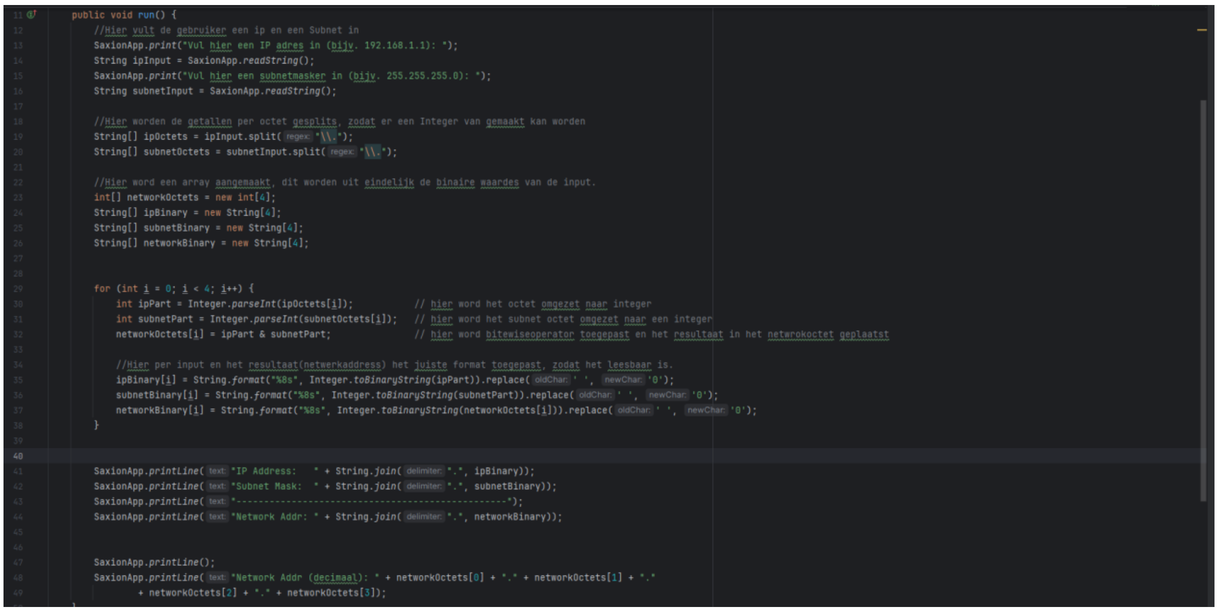
```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-------------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000
```

```
This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2⁵).
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.

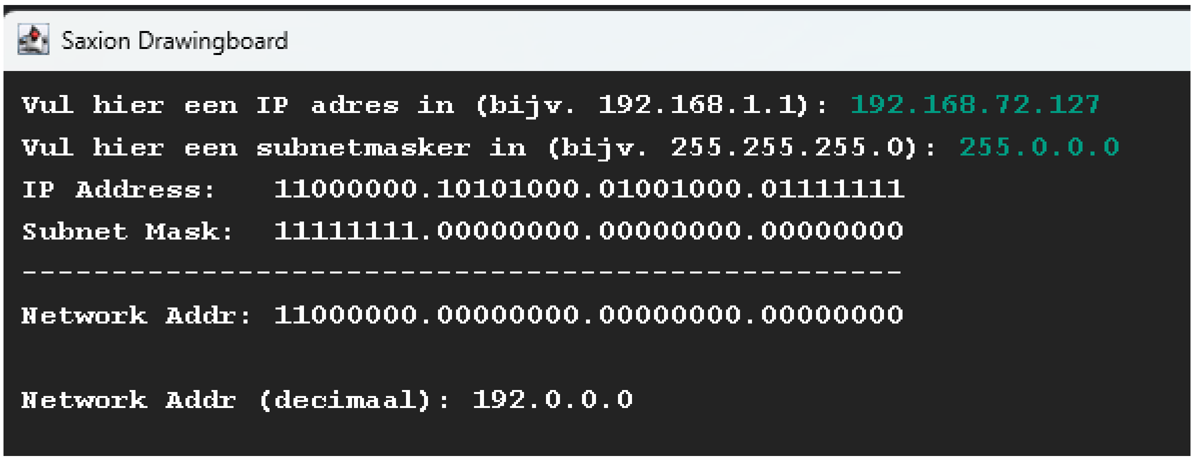
Ready? Save this file and export it as a pdf file with the name: **week6.pdf**

Screenshots bonus assignment:



```java
    public void run() {
        //Hier vult de gebruiker een ip en een Subnet in
        SaxionApp.print("Vul hier een IP adres in (bijv. 192.168.1.1): ");
        String ipInput = SaxionApp.readString();
        SaxionApp.print("Vul hier een subnetmasker in (bijv. 255.255.255.0): ");
        String subnetInput = SaxionApp.readString();

        //Hier worden de getallen per octet gesplits, zodat er een Integer van gemaakt kan worden
        String[] ipOctets = ipInput.split("\\.");
        String[] subnetOctets = subnetInput.split("\\.");

        //Hier word een array aangemaakt, dit worden uit eindelijk de binaire waardes van de input.
        int[] networkOctets = new int[4];
        String[] ipBinary = new String[4];
        String[] subnetBinary = new String[4];
        String[] networkBinary = new String[4];


        for (int i = 0; i < 4; i++) {
            int ipPart = Integer.parseInt(ipOctets[i]);           // hier word het octet omgezet naar integer
            int subnetPart = Integer.parseInt(subnetOctets[i]);   // hier word het subnet octet omgezet naar een integer
            networkOctets[i] = ipPart & subnetPart;               // hier word bitewiseoperator toegepast en het resultaat in het netwrokoctet geplaatst

            //Hier per input en het resultaat(netwerkaddress) het juiste format toegepast, zodat het leesbaar is.
            ipBinary[i] = String.format("%8s", Integer.toBinaryString(ipPart)).replace(' ', '0');
            subnetBinary[i] = String.format("%8s", Integer.toBinaryString(subnetPart)).replace(' ', '0');
            networkBinary[i] = String.format("%8s", Integer.toBinaryString(networkOctets[i])).replace(' ', '0');
        }


        SaxionApp.printLine("IP Address:   " + String.join(".", ipBinary));
        SaxionApp.printLine("Subnet Mask:  " + String.join(".", subnetBinary));
        SaxionApp.printLine("-------------------------------------------------");
        SaxionApp.printLine("Network Addr: " + String.join(".", networkBinary));


        SaxionApp.printLine();
        SaxionApp.printLine("Network Addr (decimaal): " + networkOctets[0] + "." + networkOctets[1] + "."
                + networkOctets[2] + "." + networkOctets[3]);
```

Bij dit voorbeeld ben ik gegaan voor een klasse A ip address.



```
Vul hier een IP adres in (bijv. 192.168.1.1): 192.168.72.127
Vul hier een subnetmasker in (bijv. 255.255.255.0): 255.0.0.0
IP Address:    11000000.10101000.01001000.01111111
Subnet Mask:   11111111.00000000.00000000.00000000
--------------------------------------------------
Network Addr:  11000000.00000000.00000000.00000000

Network Addr (decimaal): 192.0.0.0
```

De code:

```
import nl.saxion.app.SaxionApp;

import java.util.ArrayList;

public class Application implements Runnable {

  public static void main(String[] args) {
    SaxionApp.start(new Application(), 1024, 1024);
  }

  public void run() {
    //Hier vult de gebruiker een ip en een Subnet in
    SaxionApp.print("Vul hier een IP adres in (bijv. 192.168.1.1): ");
    String ipInput = SaxionApp.readString();
    SaxionApp.print("Vul hier een subnetmasker in (bijv. 255.255.255.0): ");
    String subnetInput = SaxionApp.readString();

    //Hier worden de getallen per octet gesplits, zodat er een Integer van gemaakt kan worden
    String[] ipOctets = ipInput.split("\\.");
    String[] subnetOctets = subnetInput.split("\\.");

    //Hier word een array aangemaakt, dit worden uit eindelijk de binaire waardes van de input.
    int[] networkOctets = new int[4];
    String[] ipBinary = new String[4];
    String[] subnetBinary = new String[4];
    String[] networkBinary = new String[4];


    for (int i = 0; i < 4; i++) {
      int ipPart = Integer.parseInt(ipOctets[i]);          // hier word het octet omgezet naar integer
      int subnetPart = Integer.parseInt(subnetOctets[i]);  // hier word het subnet octet omgezet naar
een integer
      networkOctets[i] = ipPart & subnetPart;              // hier word bitewiseoperator toegepast en
het resultaat in het netwrokoctet geplaatst
```

```
            //Hier per input en het resultaat(netwerkaddress) het juiste format toegepast, zodat het
leesbaar is.
            ipBinary[i] = String.format("%8s", Integer.toBinaryString(ipPart)).replace(' ', '0');
            subnetBinary[i] = String.format("%8s", Integer.toBinaryString(subnetPart)).replace(' ', '0');
            networkBinary[i] = String.format("%8s", Integer.toBinaryString(networkOctets[i])).replace(' ',
'0');
        }


        SaxionApp.printLine("IP Address:   " + String.join(".", ipBinary));
        SaxionApp.printLine("Subnet Mask:  " + String.join(".", subnetBinary));
        SaxionApp.printLine("------------------------------------------------");
        SaxionApp.printLine("Network Addr: " + String.join(".", networkBinary));


        SaxionApp.printLine();
        SaxionApp.printLine("Network Addr (decimaal): " + networkOctets[0] + "." + networkOctets[1] +
"."
                + networkOctets[2] + "." + networkOctets[3]);
    }



}
```