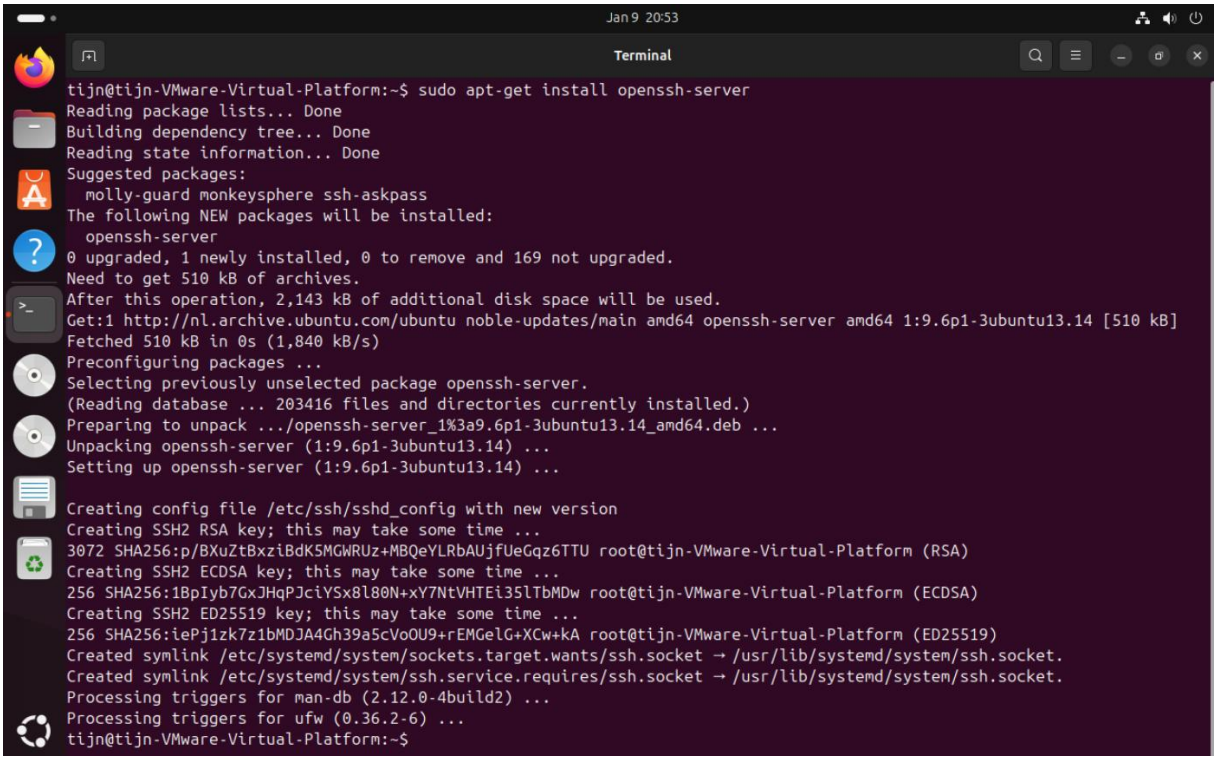
Template Week 6 – Networking

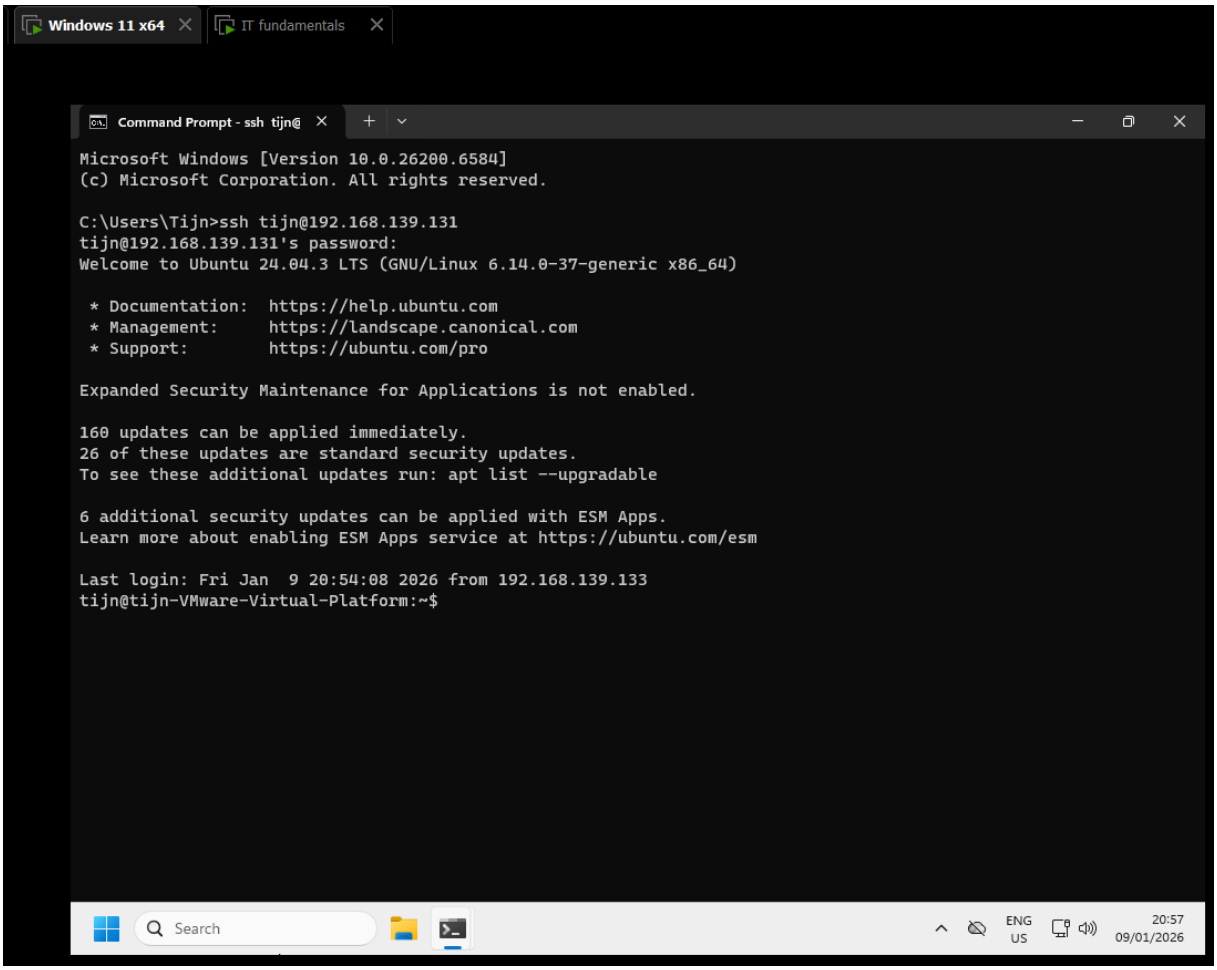
Student number: 591007

Assignment 6.1: Working from home

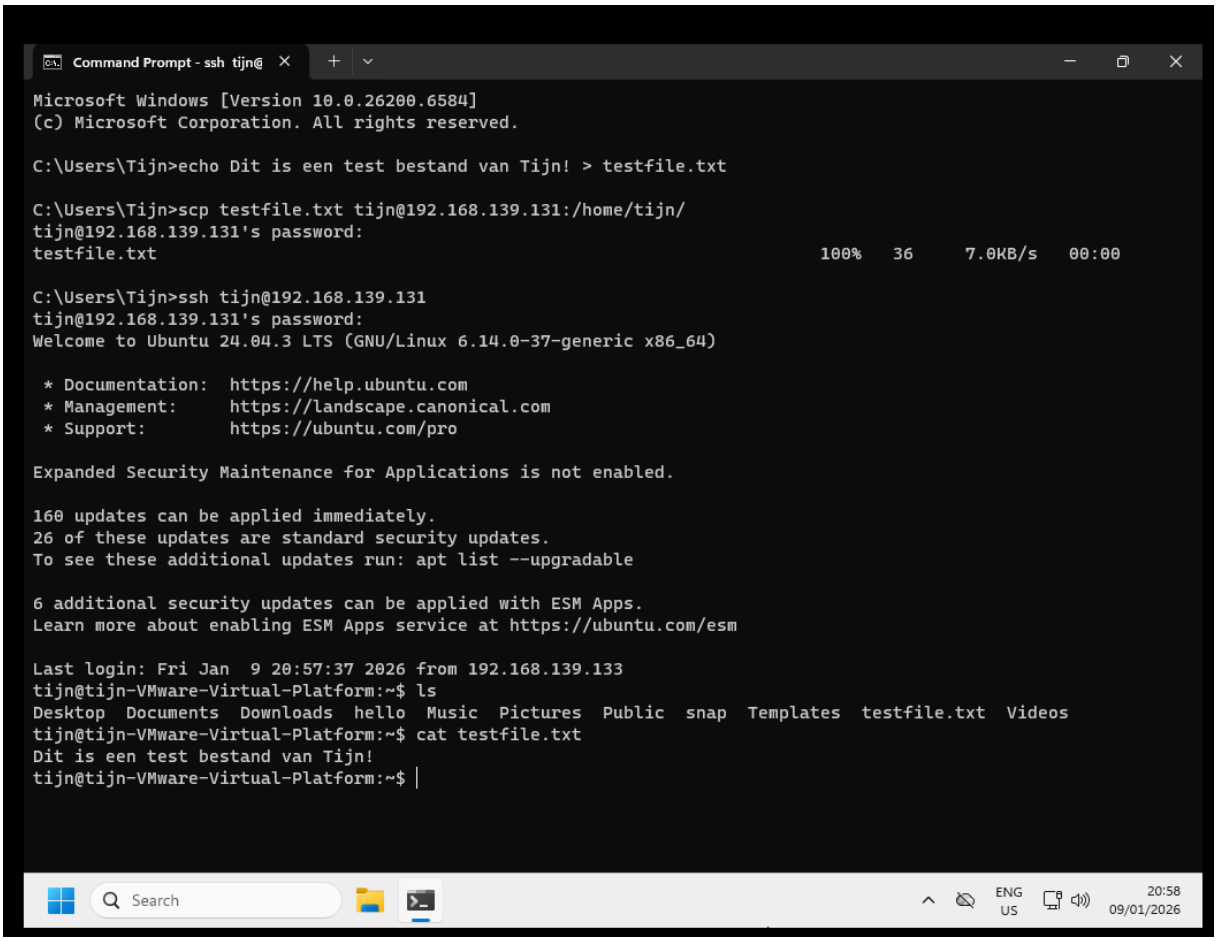
Screenshot installation openssh-server:



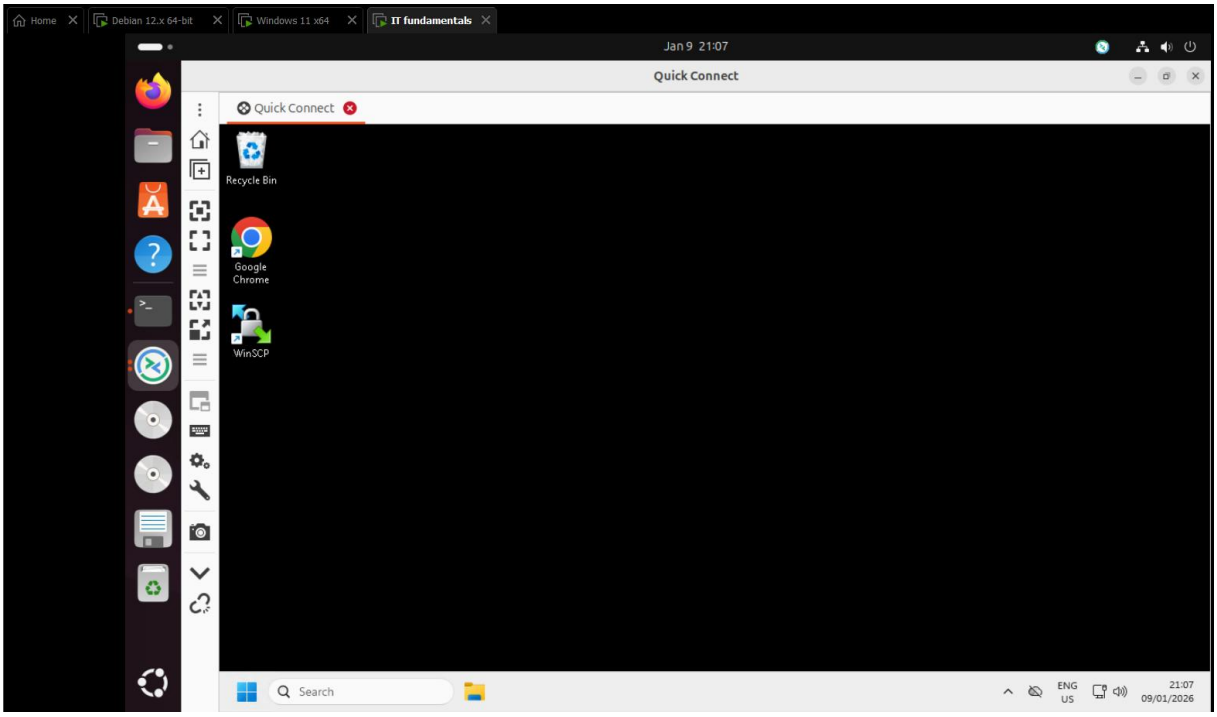
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:

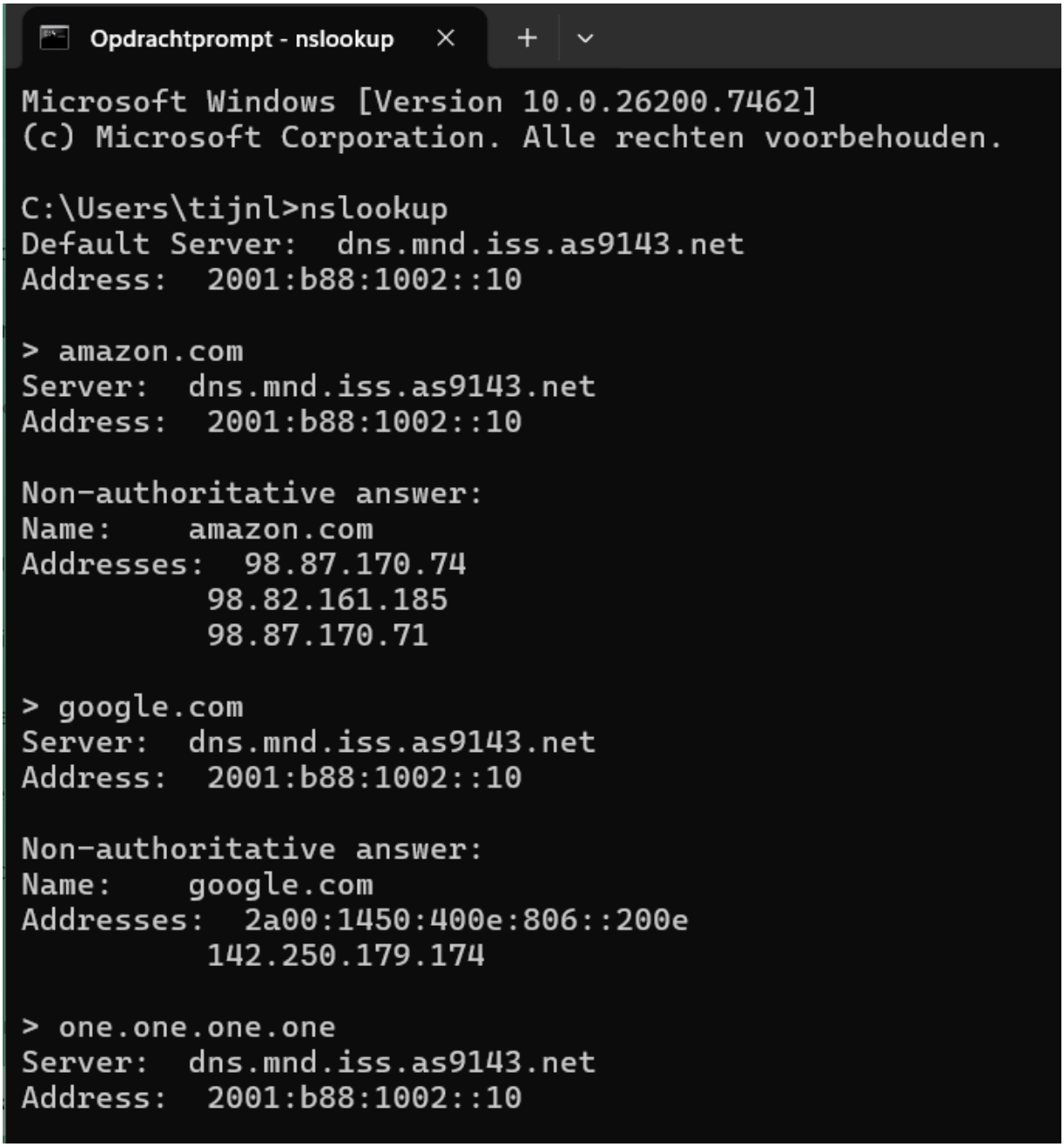


Screenshot remmina:



Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:



Screenshot website visit via IP address:

Assignment 6.3: subnetting

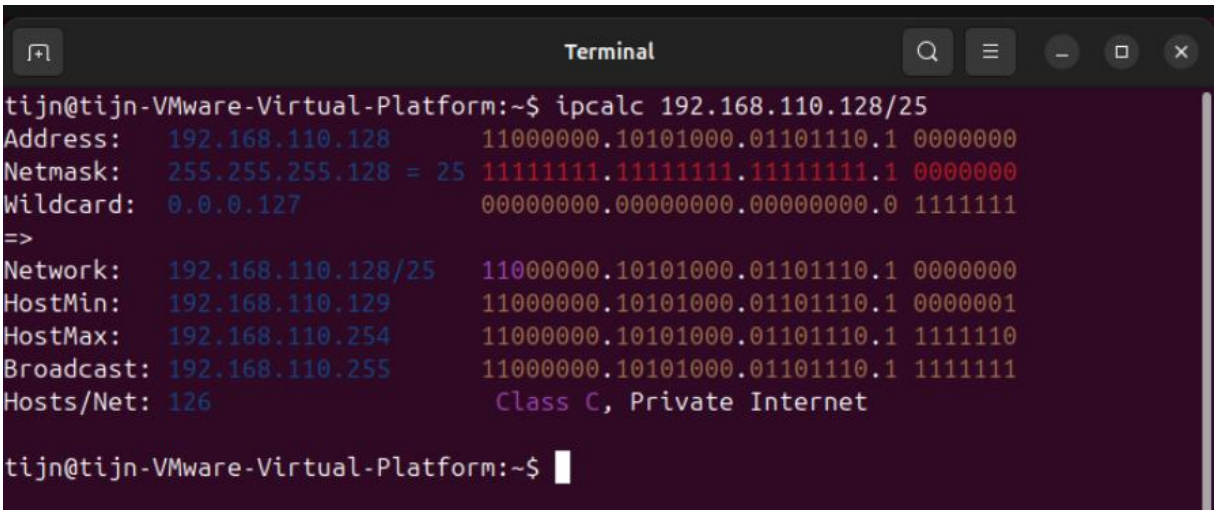
How many IP addresses are in this network configuration 192.168.110.128/25?

128 IP adressen want /25 betekent dat de eerste 25 bits voor het netwerk zijn. Dat laat dus maar 7 bits over voor unieke IP adressen en 2 tot de macht 7 is 128.

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

192.168.110.129 t/m 192.168.110.254 dit zijn er 2 minder. Dit komt doordat het eerste adres wordt gebruikt voor het netwerk en het laatste adres voor broadcasting.

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



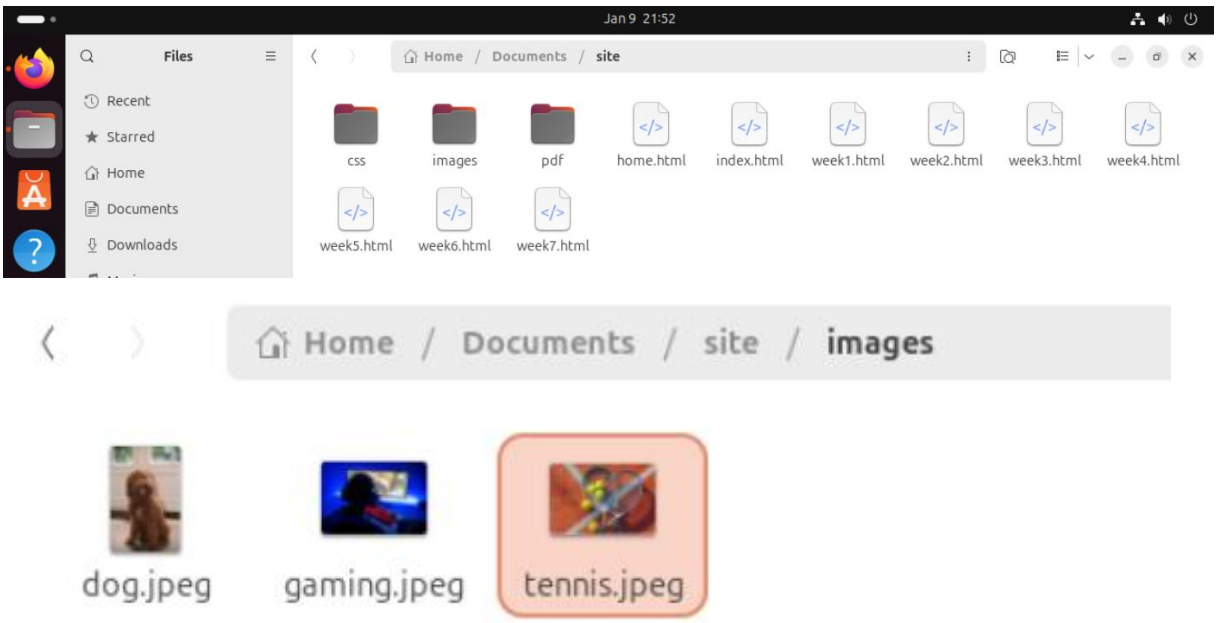
Explain the above calculation in your own words.

Assignment 6.4: HTML

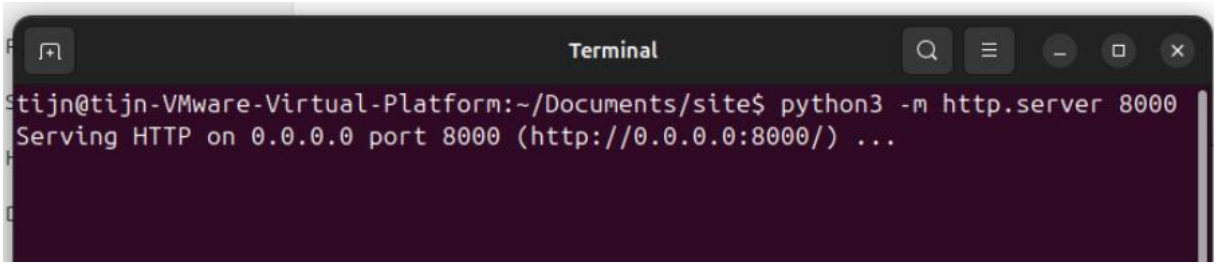
Screenshot IP address Ubuntu VM:

```
Terminal
tijn@tijn-VMware-Virtual-Platform:~/Documents/site$ 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:29:51:82:ba brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.139.131/24 brd 192.168.139.255 scope global dynamic noprefixrou
te ens33
       valid_lft 1737sec preferred_lft 1737sec
    inet6 fe80::20c:29ff:fe51:82ba/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
tijn@tijn-VMware-Virtual-Platform:~/Documents/site$
```

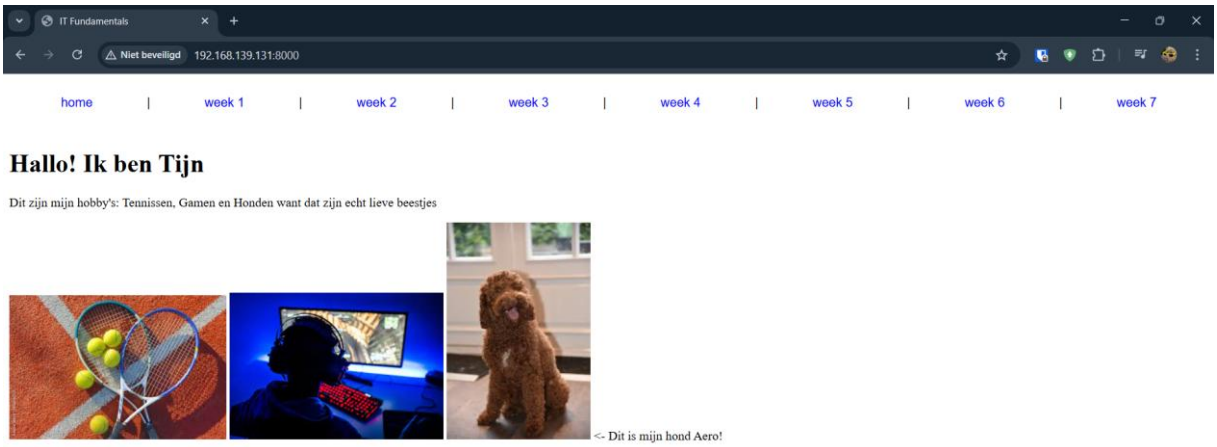
Screenshot of Site directory contents:



Screenshot python3 webserver command:



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.
```

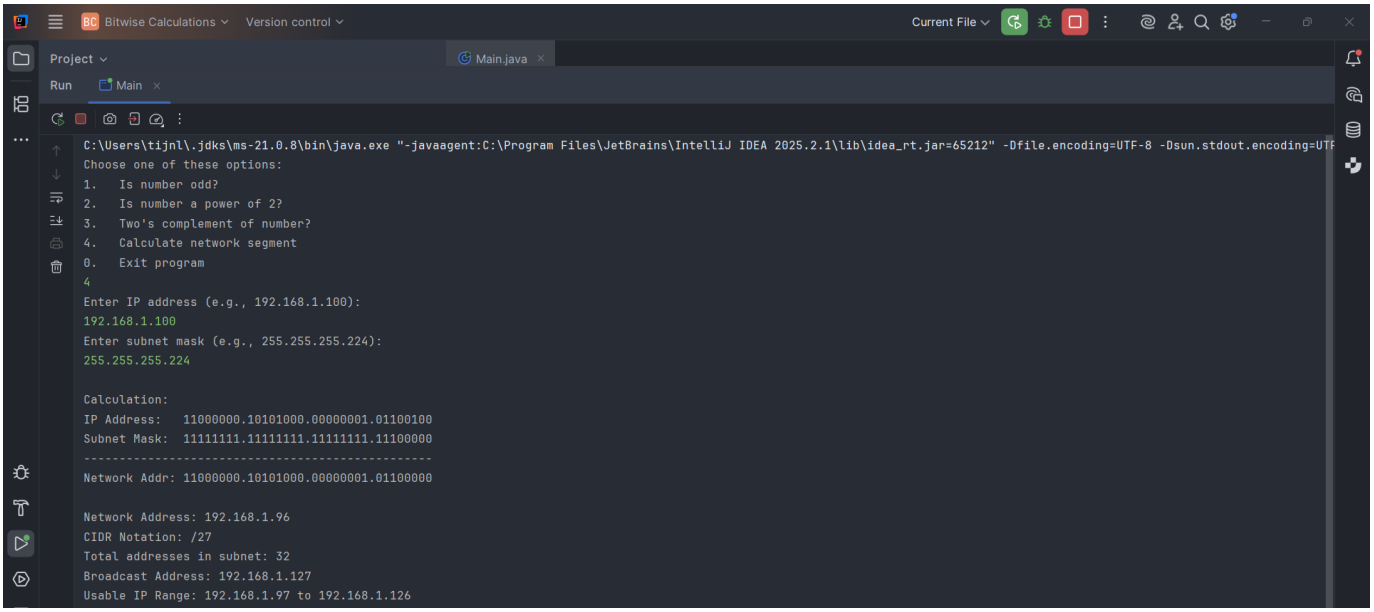
Paste source code here, with a screenshot of a working application.

```
import java.util.Scanner;

public class Main {
    public static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        while (true) {
            printMainMenu();

            int input = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (input) {
                case 1:
                    printOddNumber();
                    break;
                case 2:
                    printPowerOf2();
                    break;
                case 3:
                    printTwosComplement();
                    break;
                case 4:
                    calculateNetworkSegment();
                    break;
                case 0:
                    return;
            }
        }
    }

    public static void printMainMenu() {
        System.out.println("Choose one of these options:");
        System.out.println("1.   Is number odd?");
        System.out.println("2.   Is number a power of 2?");
        System.out.println("3.   Two's complement of number?");
        System.out.println("4.   Calculate network segment");
        System.out.println("0.   Exit program");
    }

    public static void printOddNumber() {
        System.out.println("Fill in the number you want to check: ");

        int number = scanner.nextInt();

        if ((number & 1) == 1) {
            System.out.println(number + " is odd");
        } else {
```

```
        System.out.println(number + " is even");
    }
}

public static void printPowerOf2() {
    System.out.println("Fill in the number you want to check: ");

    int number = scanner.nextInt();

    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 printTwosComplement() {
    System.out.println("Fill in the number you want to convert: ");

    int number = scanner.nextInt();

    int twosComplement = ~number + 1;

    System.out.println("Two's complement of " + number + " is " + twosComplement);
}

public static void calculateNetworkSegment() {
    System.out.println("Enter IP address (e.g., 192.168.1.100): ");
    String ipAddress = scanner.nextLine();

    System.out.println("Enter subnet mask (e.g., 255.255.255.224): ");
    String subnetMask = scanner.nextLine();

    // Convert IP address string to array of integers
    String[] ipParts = ipAddress.split("\\.");
    int[] ip = new int[4];
    for (int i = 0; i < 4; i++) {
        ip[i] = Integer.parseInt(ipParts[i]);
    }

    // Convert subnet mask string to array of integers
    String[] maskParts = subnetMask.split("\\.");
    int[] mask = new int[4];
    for (int i = 0; i < 4; i++) {
        mask[i] = Integer.parseInt(maskParts[i]);
    }

    // Calculate network address using bitwise AND
```

```
int[] networkAddress = new int[4];
for (int i = 0; i < 4; i++) {
   networkAddress[i] = ip[i] & mask[i];
}

// Print results in binary format
System.out.println("\nCalculation:");
System.out.println("IP Address:   " + toBinaryString(ip));
System.out.println("Subnet Mask:  " + toBinaryString(mask));
System.out.println("-----------------------------------------------");
System.out.println("Network Addr: " + toBinaryString(networkAddress));

// Print network address in decimal
System.out.println("\nNetwork Address: " + networkAddress[0] + "." +
      networkAddress[1] + "." + networkAddress[2] + "." +
      networkAddress[3]);

// Calculate CIDR notation (count 1's in subnet mask)
int cidr = countOnes(mask);
System.out.println("CIDR Notation: /" + cidr);

// Calculate number of addresses in subnet
int hostBits = 32 - cidr;
int totalAddresses = (int) Math.pow(2, hostBits);
System.out.println("Total addresses in subnet: " + totalAddresses);

// Calculate broadcast address
int[] broadcastAddress = new int[4];
for (int i = 0; i < 4; i++) {
   broadcastAddress[i] = networkAddress[i] | (~mask[i] & 0xFF);
}

System.out.println("Broadcast Address: " + broadcastAddress[0] + "." +
      broadcastAddress[1] + "." + broadcastAddress[2] + "." +
      broadcastAddress[3]);

// Calculate usable range
int[] firstUsable = networkAddress.clone();
firstUsable[3] += 1;

int[] lastUsable = broadcastAddress.clone();
lastUsable[3] -= 1;

System.out.println("Usable IP Range: " + firstUsable[0] + "." +
      firstUsable[1] + "." + firstUsable[2] + "." +
      firstUsable[3] + " to " + lastUsable[0] + "." +
      lastUsable[1] + "." + lastUsable[2] + "." +
      lastUsable[3]);
```

```
        System.out.println();
    }

    // Convert IP address octets to binary string format
    public static String toBinaryString(int[] octets) {
        StringBuilder result = new StringBuilder();
        for (int i = 0; i < octets.length; i++) {
            String binary = String.format("%8s", Integer.toBinaryString(octets[i]))
                    .replace(' ', '0');
            result.append(binary);
            if (i < octets.length - 1) {
                result.append(".");
            }
        }
        return result.toString();
    }

    // Count number of 1's in subnet mask (for CIDR notation)
    public static int countOnes(int[] mask) {
        int count = 0;
        for (int octet : mask) {
            count += Integer.bitCount(octet);
        }
        return count;
    }
}
```



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