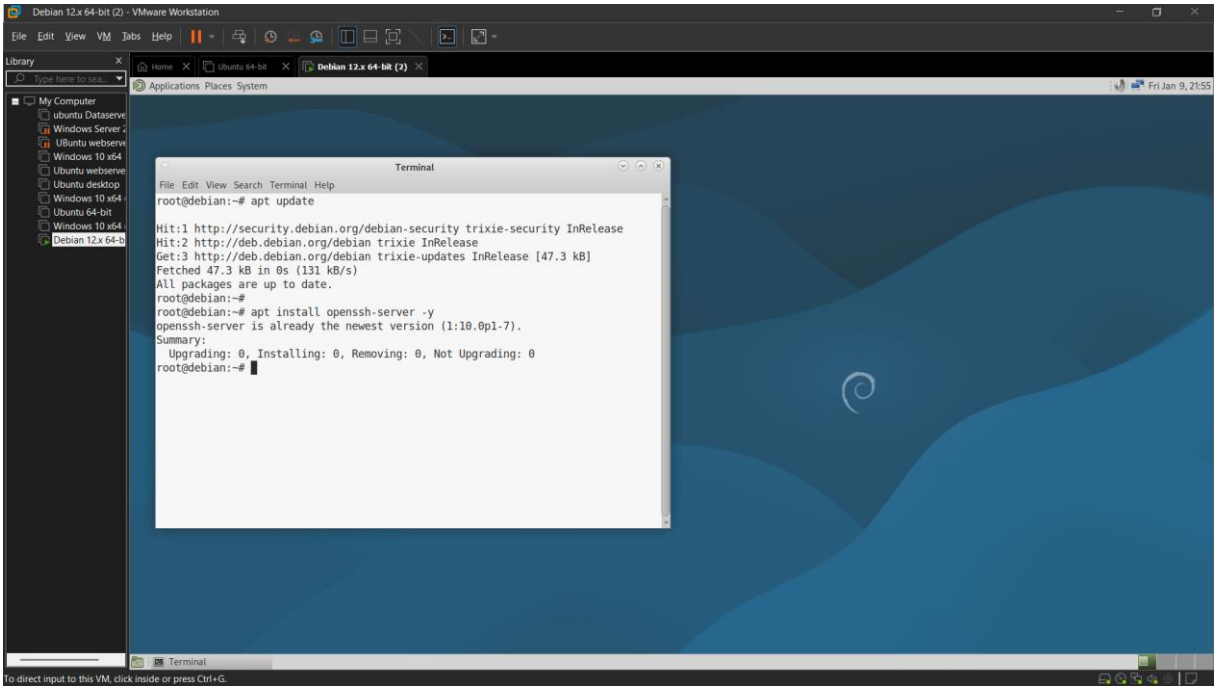
Template Week 6 – Networking

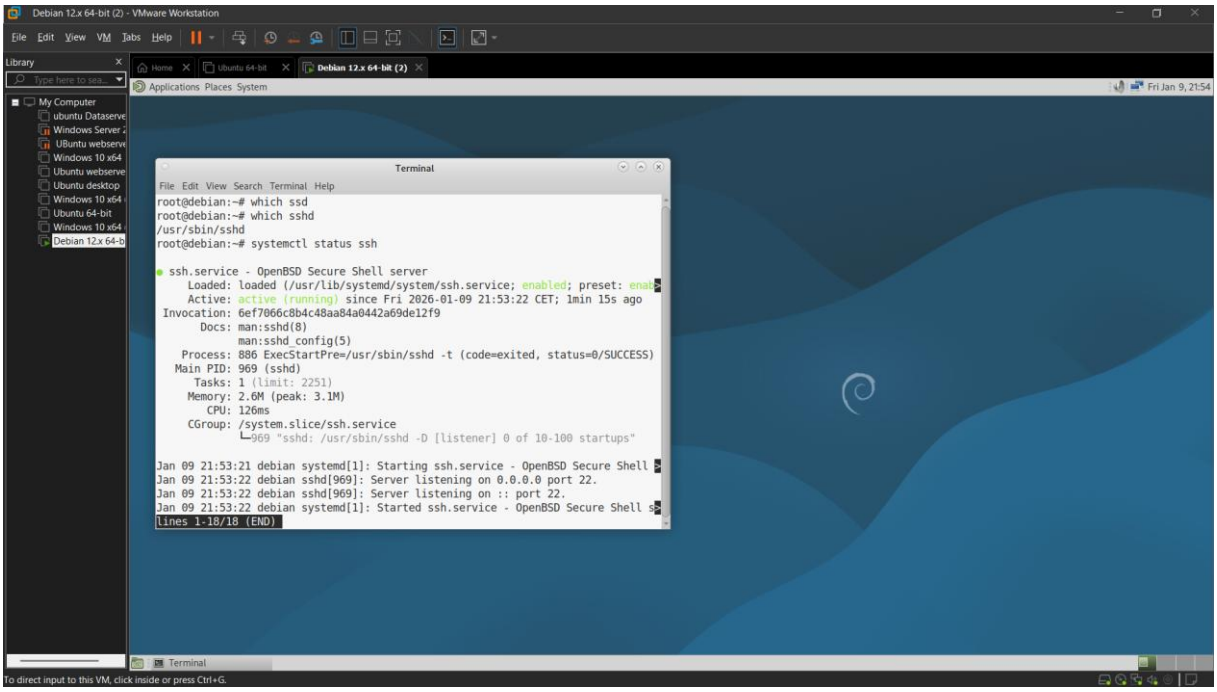
Student number: 528668

Assignment 6.1: Working from home

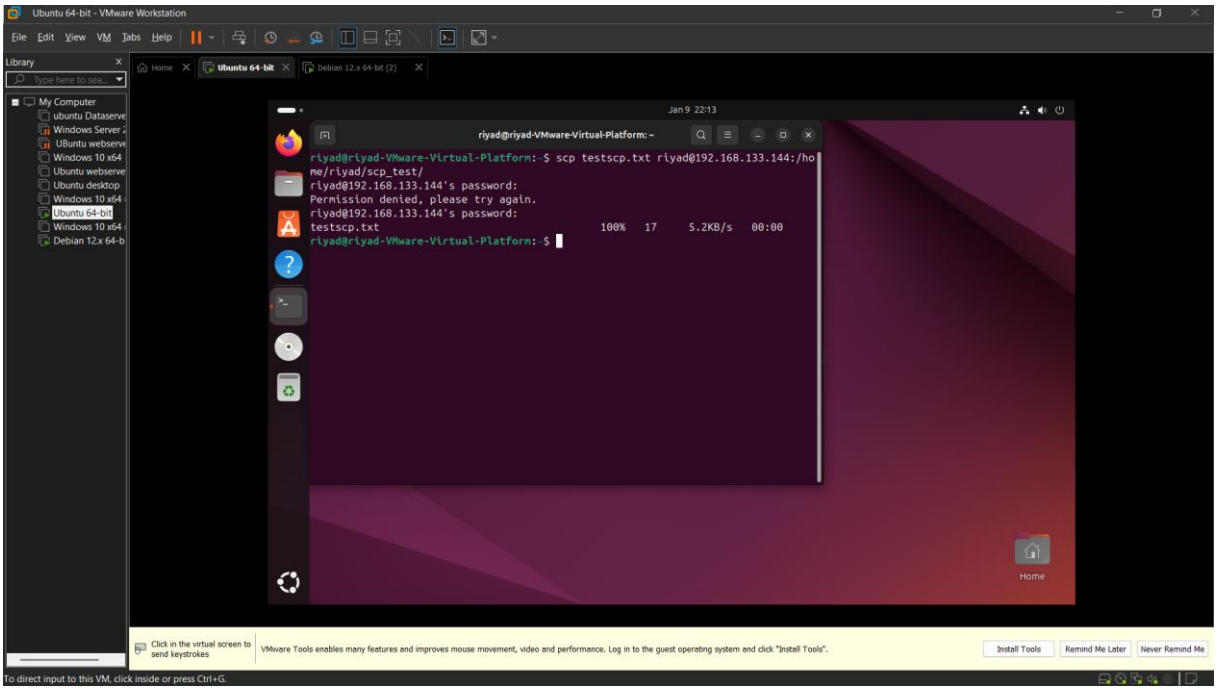
Screenshot installation openssh-server:



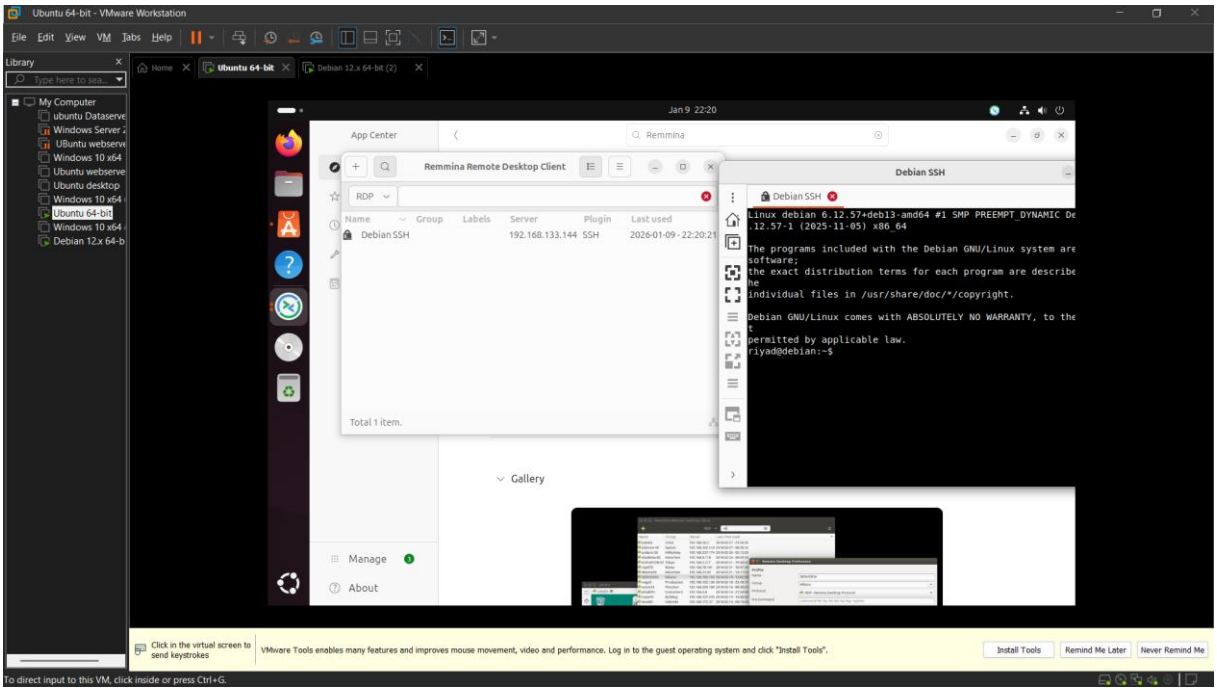
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:

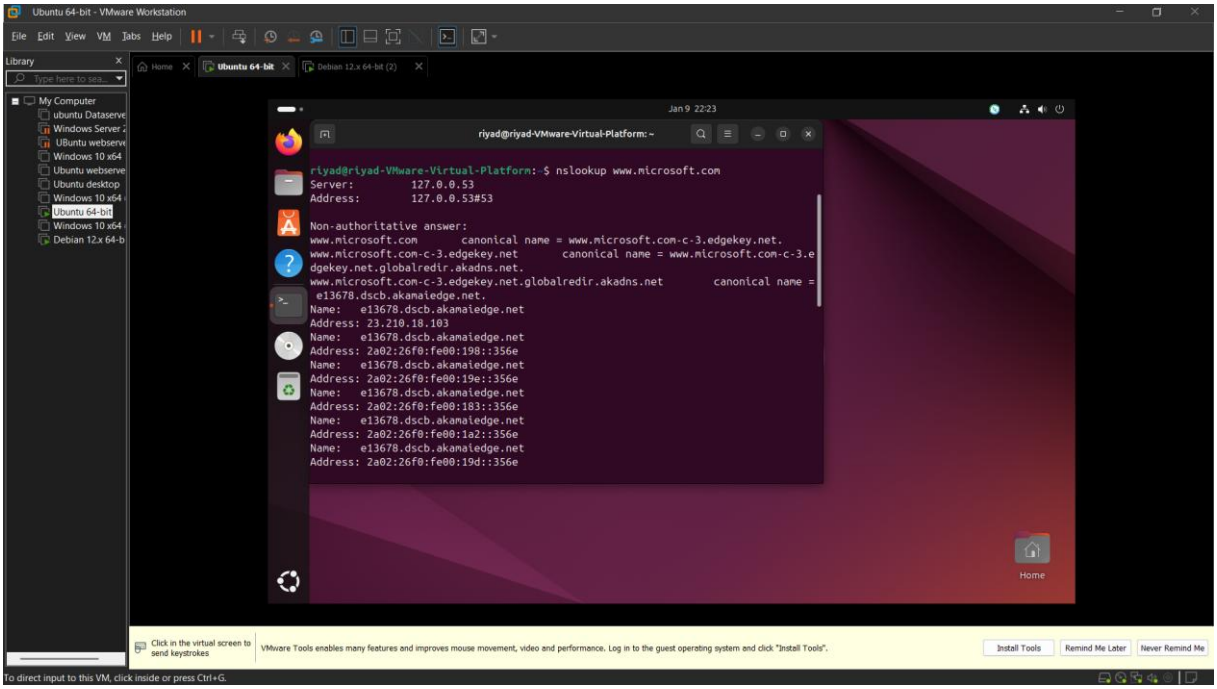


Screenshot remmina:

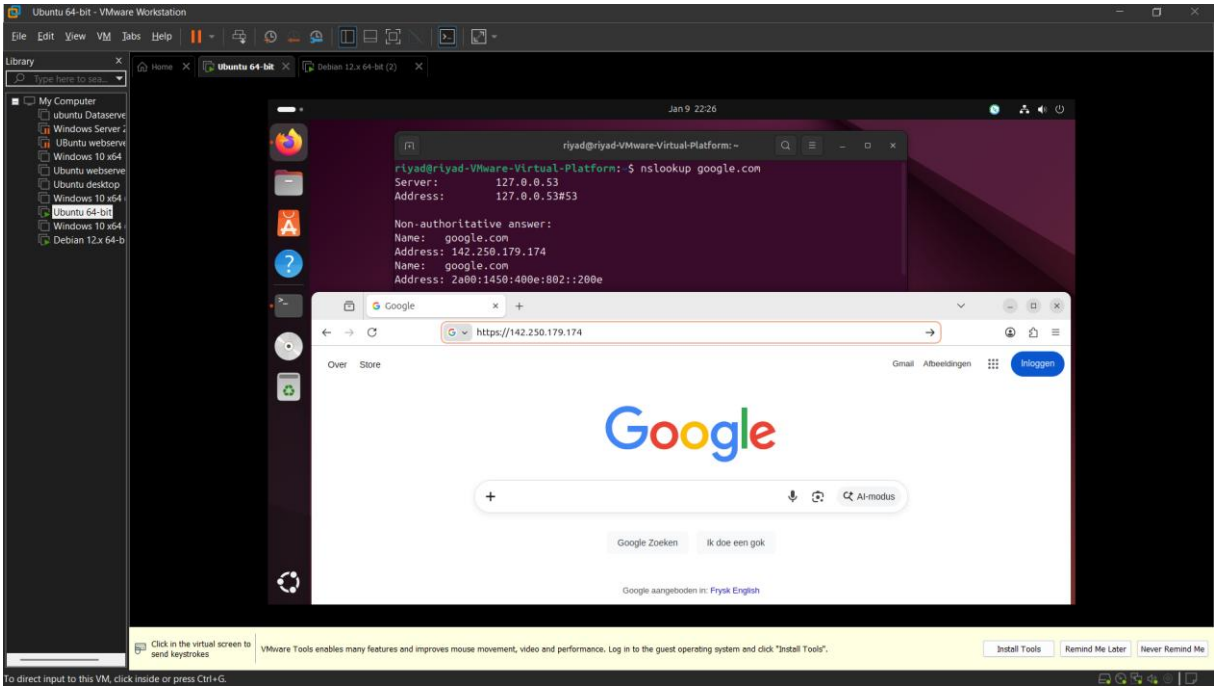


Assignment 6.2: IP addresses websites

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?

Een /25 betekent: 32(totaal aantal bits) − 25 = 7 host-bits.
Aantal IP-adressen = $2^7 = 128$

Dus 128 IP adressen

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

Bij een subnet zijn altijd 2 adressen “gereserveerd”:

Network address (eerste adres)

Broadcast address (laatste adres)

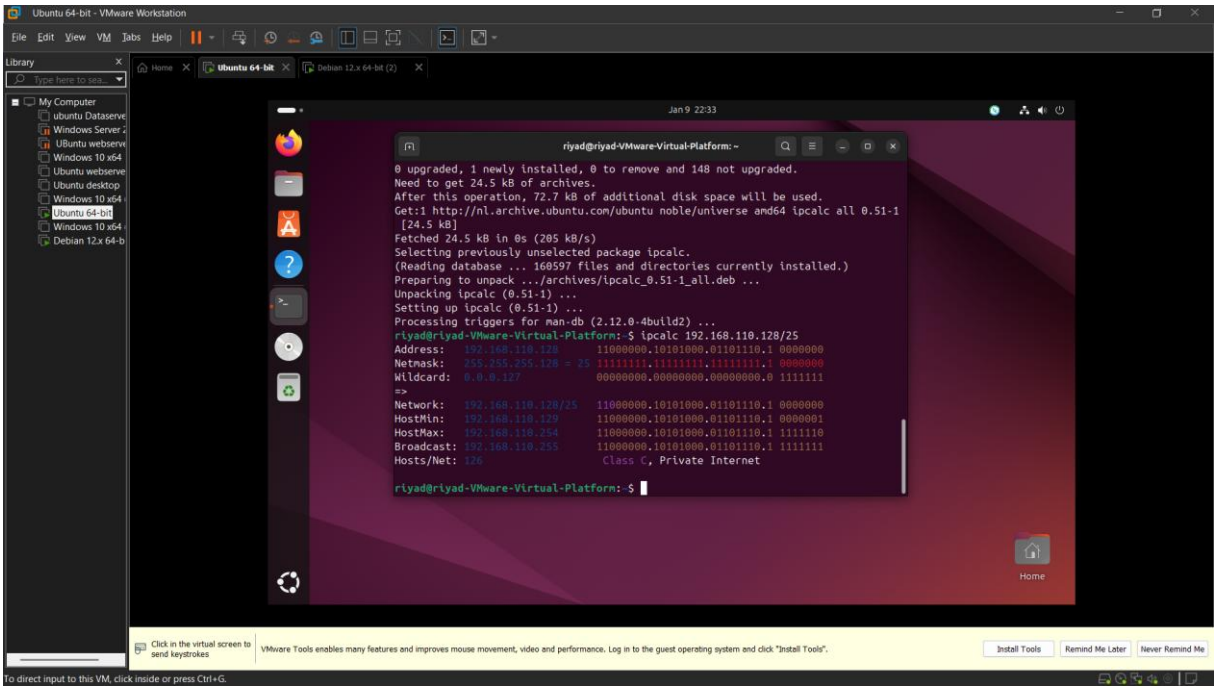
Voor 192.168.110.128/25:

Network = 192.168.110.128

Broadcast = 192.168.110.255

Dus usuble hosts is 255-128 = 126

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

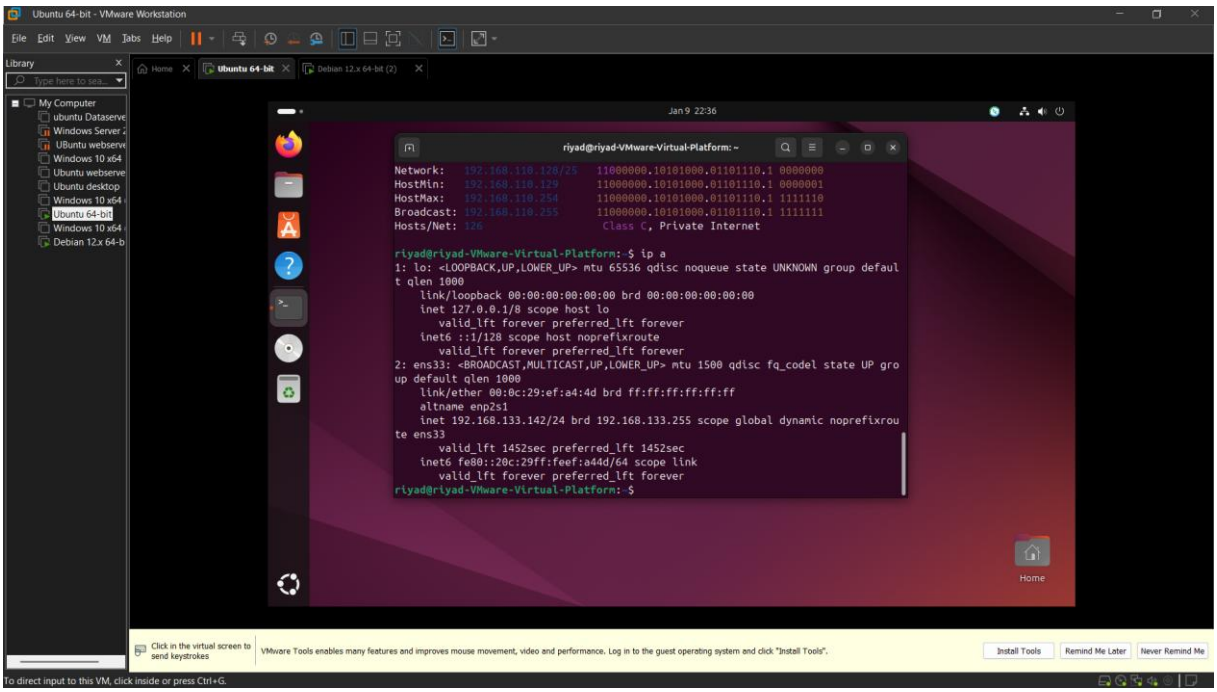


Explain the above calculation in your own words.

Een /25 subnet betekent dat de eerste 25 bits van het IP-adres het netwerkdeel zijn en dat er 7 bits overblijven voor hosts. Met 7 host-bits kun je $2^7 = 128$adressen maken in dat subnet.
Het eerste adres (192.168.110.128) is het netwerkadres en wordt gebruikt om het netwerk zelf aan te duiden. Het laatste adres (192.168.110.255) is het broadcastadres, dat gebruikt wordt om naar alle apparaten in het subnet tegelijk te sturen.
Daarom kun je alleen de adressen ertussen uitdelen aan computers: 192.168.110.129 tot en met 192.168.110.254 (in totaal 126 bruikbare adressen).

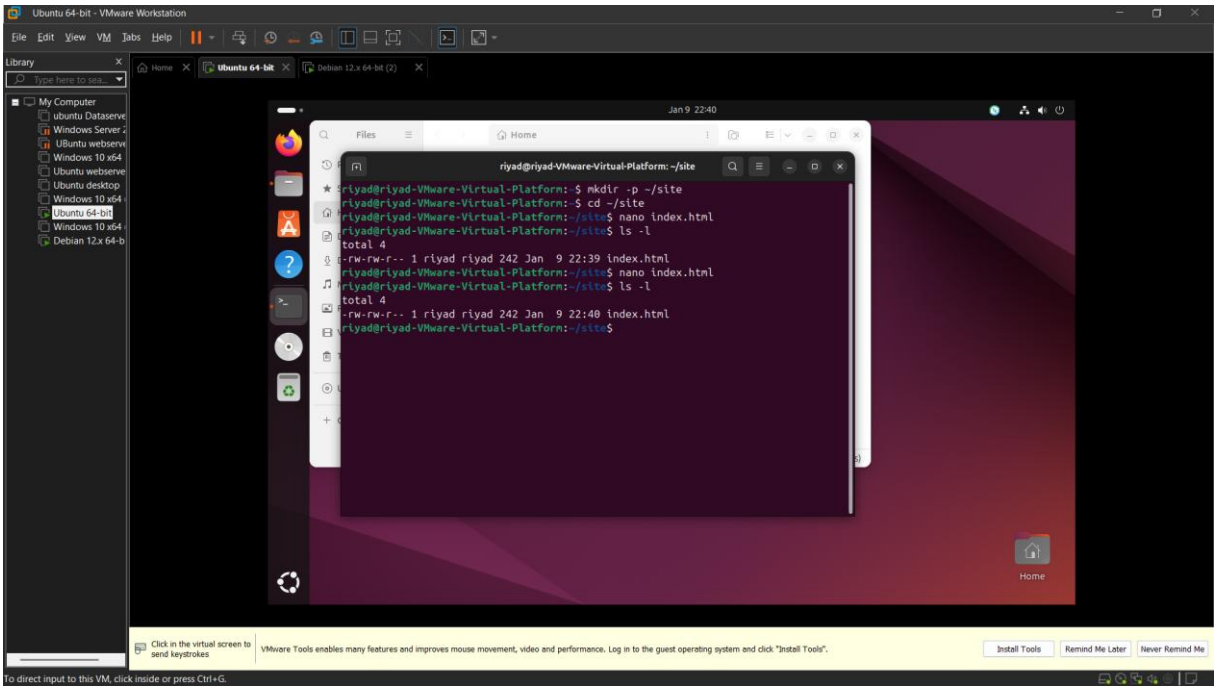
Assignment 6.4: HTML

Screenshot IP address Ubuntu VM:

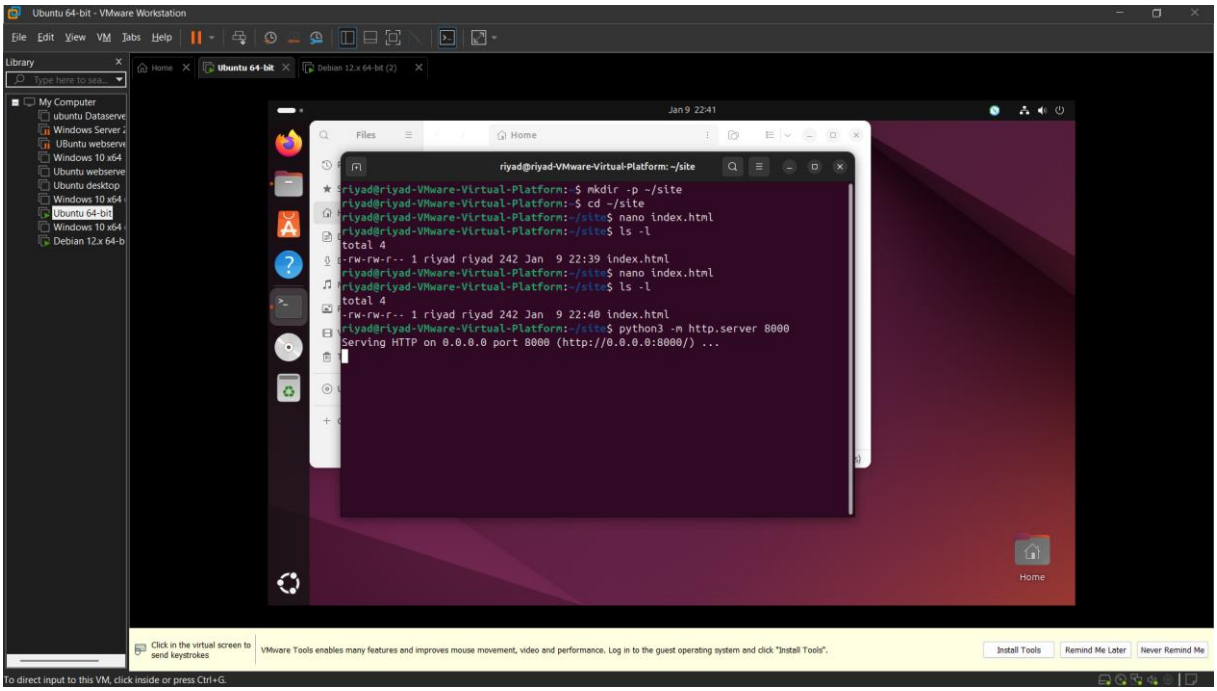


Ip address: 192.168.142/24

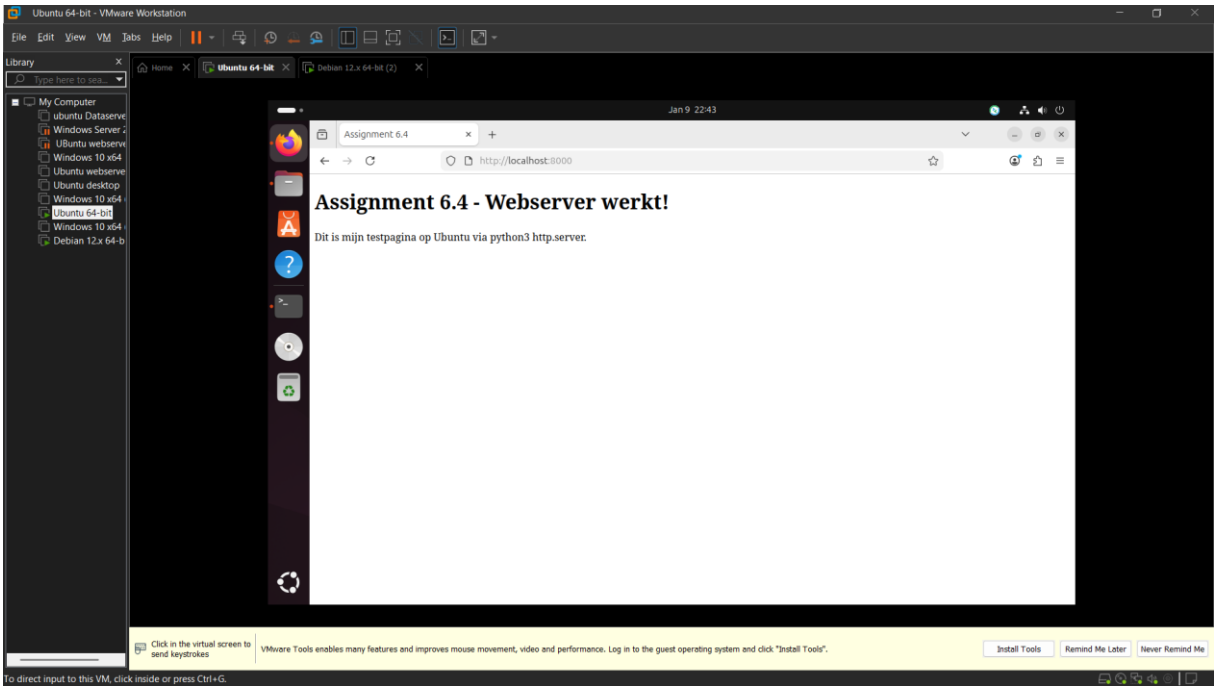
Screenshot of Site directory contents:



Screenshot python3 webserver command:



Screenshot web browser visits your site



Assignment 6.5: Network segment

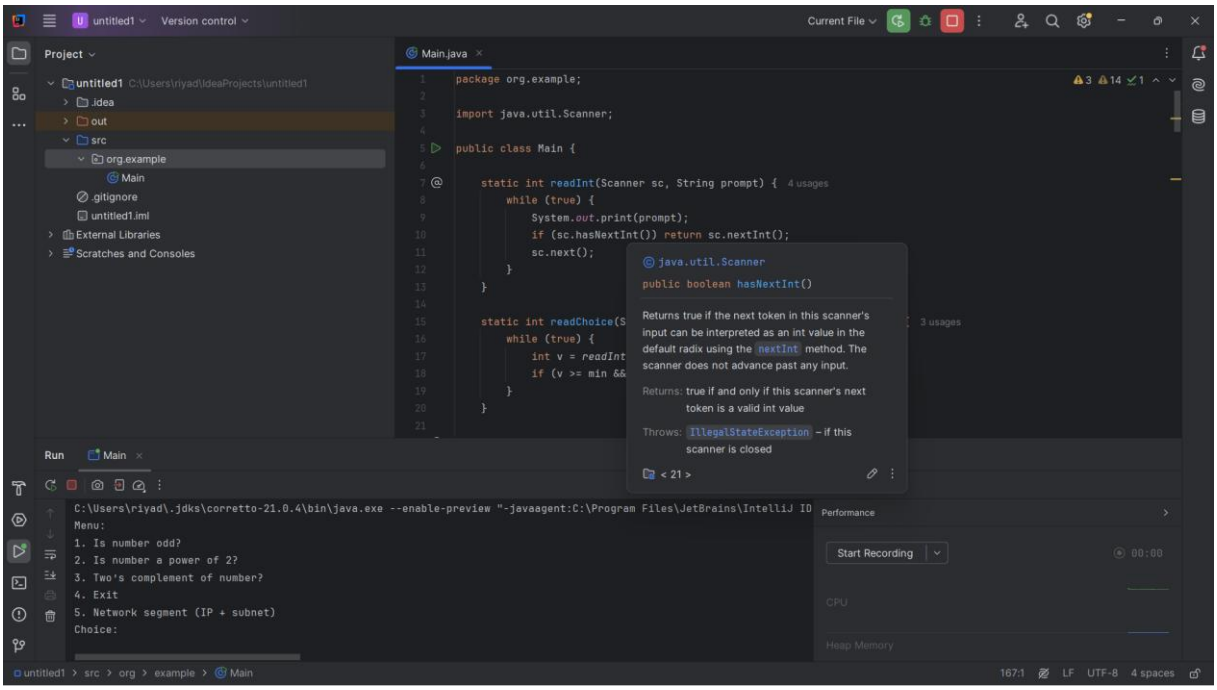
Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

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 (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.




```
package org.example;

import java.util.Scanner;

public class Main {

    static int readInt(Scanner sc, String prompt) {
        while (true) {
            System.out.print(prompt);
            if (sc.hasNextInt()) return sc.nextInt();
            sc.next();
        }
    }

    static int readChoice(Scanner sc, String prompt, int min, int max) {
        while (true) {
            int v = readInt(sc, prompt);
            if (v >= min && v <= max) return v;
        }
```

```
}

static String readLine(Scanner sc, String prompt) {
    System.out.print(prompt);
    String line = sc.nextLine();
    while (line.trim().isEmpty()) line = sc.nextLine();
    return line.trim();
}

static boolean isOdd(int n) {
    return (n & 1) == 1;
}

static boolean isPowerOfTwo(int n) {
    return n > 0 && (n & (n - 1)) == 0;
}

static long maskForBits(int bits) {
    if (bits == 32) return 0xFFFFFFFFL;
    return (1L << bits) - 1;
}

static String toBinaryPadded(long v, int bits) {
    String s = Long.toBinaryString(v);
    if (s.length() > bits) s = s.substring(s.length() - bits);
    while (s.length() < bits) s = "0" + s;
    return s;
}

static long twosComplement(long value, int bits) {
    long mask = maskForBits(bits);
```

```
    return (~value + 1) & mask;
}

static void runOdd(Scanner sc) {
    int n = readInt(sc, "Enter number: ");
    System.out.println(isOdd(n) ? "number is odd" : "number is even");
}

static void runPowerOfTwo(Scanner sc) {
    int n = readInt(sc, "Enter number: ");
    System.out.println(isPowerOfTwo(n) ? "number is a power of 2" : "number isn't a power of 2");
}

static void runTwosComplement(Scanner sc) {
    int n = readInt(sc, "Enter number: ");
    int bits = readChoice(sc, "Bit-width (8/16/32): ", 1, 32);
    while (bits != 8 && bits != 16 && bits != 32) {
        bits = readChoice(sc, "Bit-width (8/16/32): ", 1, 32);
    }
    long mask = maskForBits(bits);
    long value = ((long) n) & mask;
    long tc = twosComplement(value, bits);
    System.out.println("number: " + toBinaryPadded(value, bits));
    System.out.println("two's complement: " + toBinaryPadded(tc, bits));
}

// ===== Assignment 6.5 helpers =====

static int ipv4ToInt(String ip) {
    String[] parts = ip.split("\\.");
    if (parts.length != 4) throw new IllegalArgumentException("Invalid IP address");
```

```
    int result = 0;
    for (String p : parts) {
        int octet = Integer.parseInt(p);
        if (octet < 0 || octet > 255) throw new IllegalArgumentException("Invalid IP octet: " + octet);
        result = (result << 8) | octet;
    }
    return result;
}

static String intToIpv4(int value) {
    return ((value >>> 24) & 255) + "." +
           ((value >>> 16) & 255) + "." +
           ((value >>> 8) & 255) + "." +
           (value & 255);
}

static String toBinaryIpv4(int value) {
    String b1 = toBinaryPadded((value >>> 24) & 255, 8);
    String b2 = toBinaryPadded((value >>> 16) & 255, 8);
    String b3 = toBinaryPadded((value >>> 8) & 255, 8);
    String b4 = toBinaryPadded(value & 255, 8);
    return b1 + "." + b2 + "." + b3 + "." + b4;
}

static int prefixFromMask(int mask) {
    // Count leading 1-bits (assumes valid subnet mask)
    int prefix = 0;
    for (int i = 31; i >= 0; i--) {
        if (((mask >>> i) & 1) == 1) prefix++;
        else break;
    }
```

```
    return prefix;
}

static void runNetworkSegment(Scanner sc) {
    sc.nextLine(); // consume leftover newline from previous nextInt()
    String ipStr = readLine(sc, "Enter IP address (e.g. 192.168.1.100): ");
    String maskStr = readLine(sc, "Enter subnet mask (e.g. 255.255.255.224): ");

    int ip = ipv4ToInt(ipStr);
    int mask = ipv4ToInt(maskStr);

    // Bitwise AND for network address (kern van de opdracht)
    int network = ip & mask;

    int prefix = prefixFromMask(mask);
    int hostBits = 32 - prefix;
    int blockSize = 1 << hostBits;       // /27 -> 32
    int broadcast = network + blockSize - 1;

    System.out.println();
    System.out.println("Example: " + ipStr + "/" + prefix);
    System.out.println("Calculate the network segment");
    System.out.println("IP Address:   " + toBinaryIpv4(ip));
    System.out.println("Subnet Mask:  " + toBinaryIpv4(mask));
    System.out.println("-----------------------------------------------");
    System.out.println("Network Addr: " + toBinaryIpv4(network));
    System.out.println();
    System.out.println("Network address (decimal): " + intToIpv4(network));
    System.out.println("Broadcast address:        " + intToIpv4(broadcast));
    System.out.println("Block size (IPs):         " + blockSize);
```

```
        System.out.println("Range of this segment:     " + intToIpv4(network) + " to " +
intToIpv4(broadcast));
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("Menu:");
            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. Exit");
            System.out.println("5. Network segment (IP + subnet)"); // NEW

            int choice = readChoice(sc, "Choice: ", 1, 5);
            if (choice == 4) break;
            if (choice == 1) runOdd(sc);
            if (choice == 2) runPowerOfTwo(sc);
            if (choice == 3) runTwosComplement(sc);
            if (choice == 5) runNetworkSegment(sc);

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

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