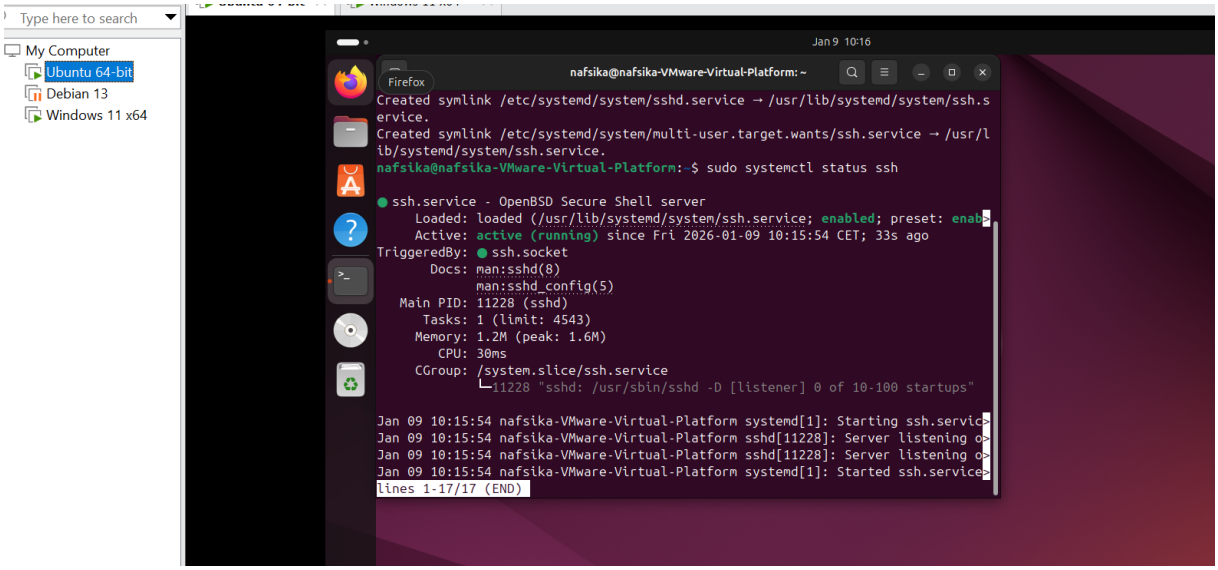
Template Week 6 – Networking

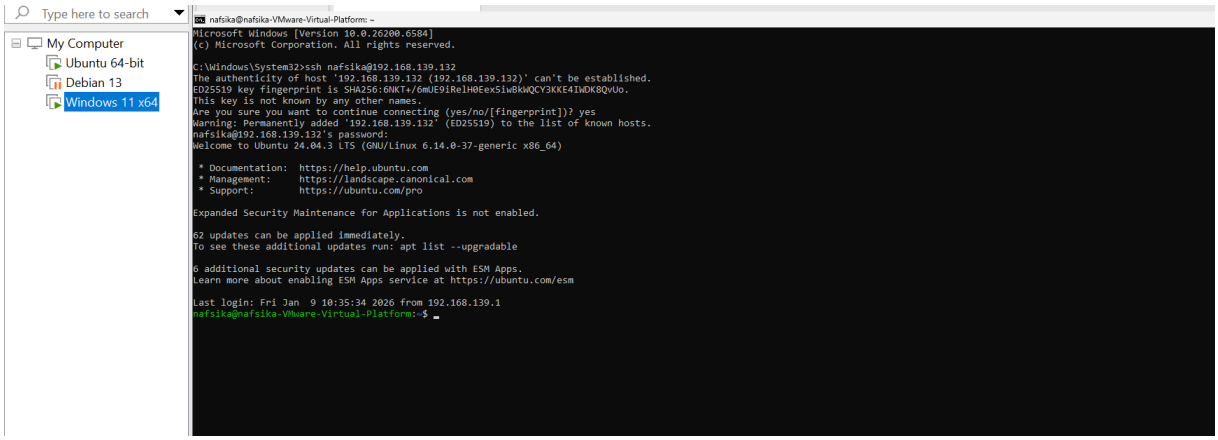
Student number: 579185 – Nafsika Pagkali

Assignment 6.1: Working from home

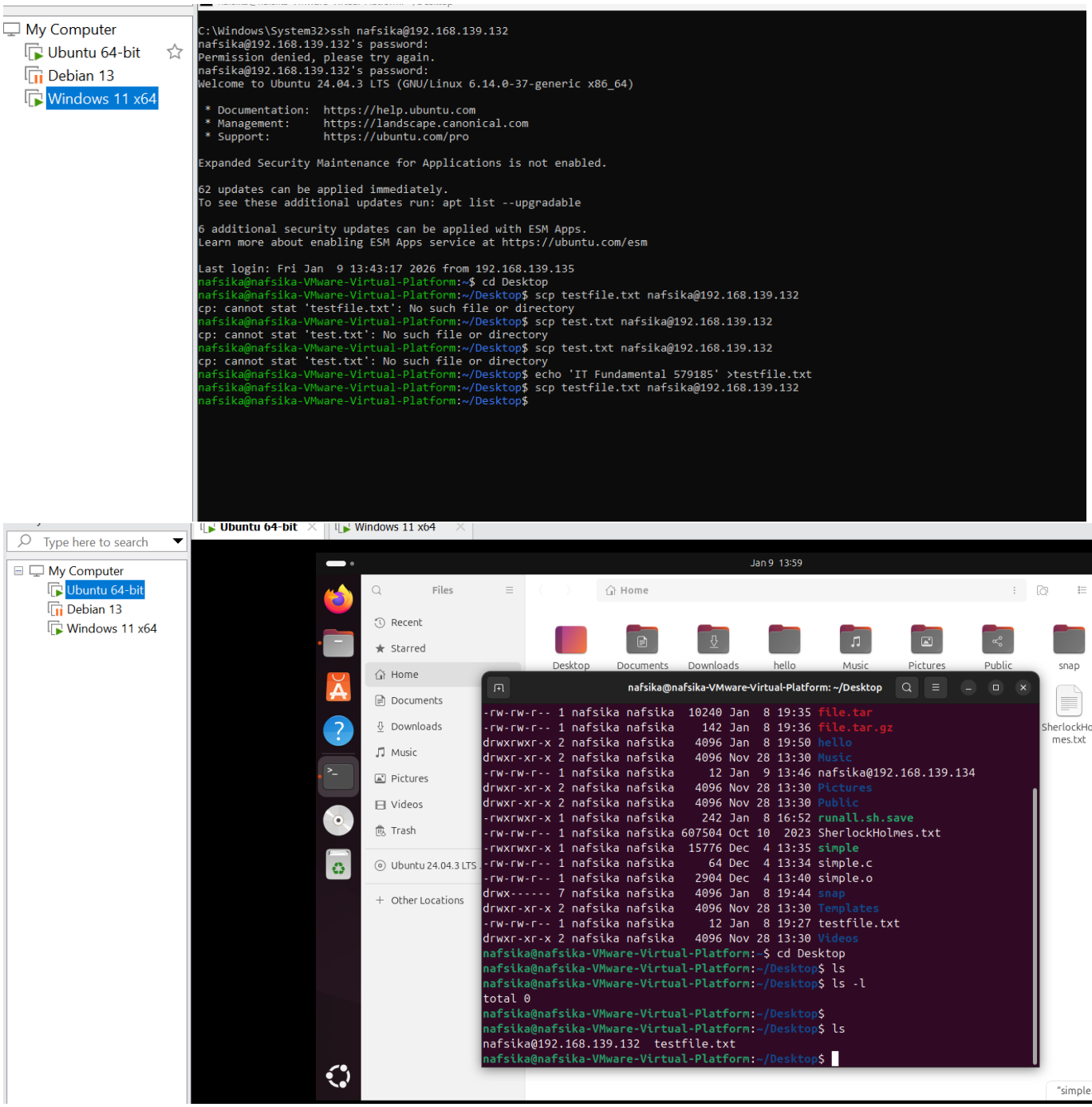
Screenshot installation openssh-server:



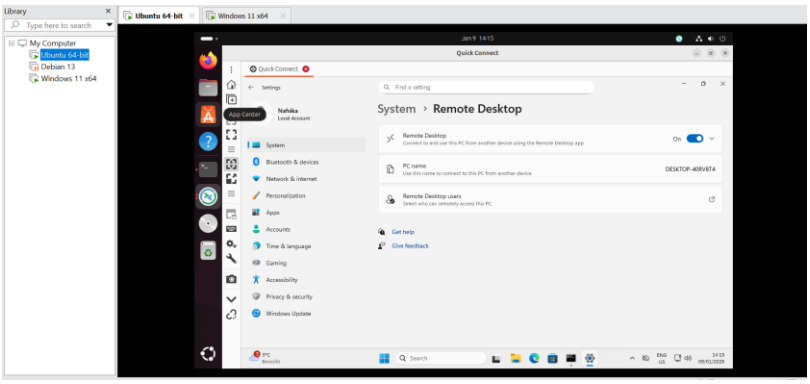
Screenshot successful SSH command execution:



Screenshot successful execution SCP command:

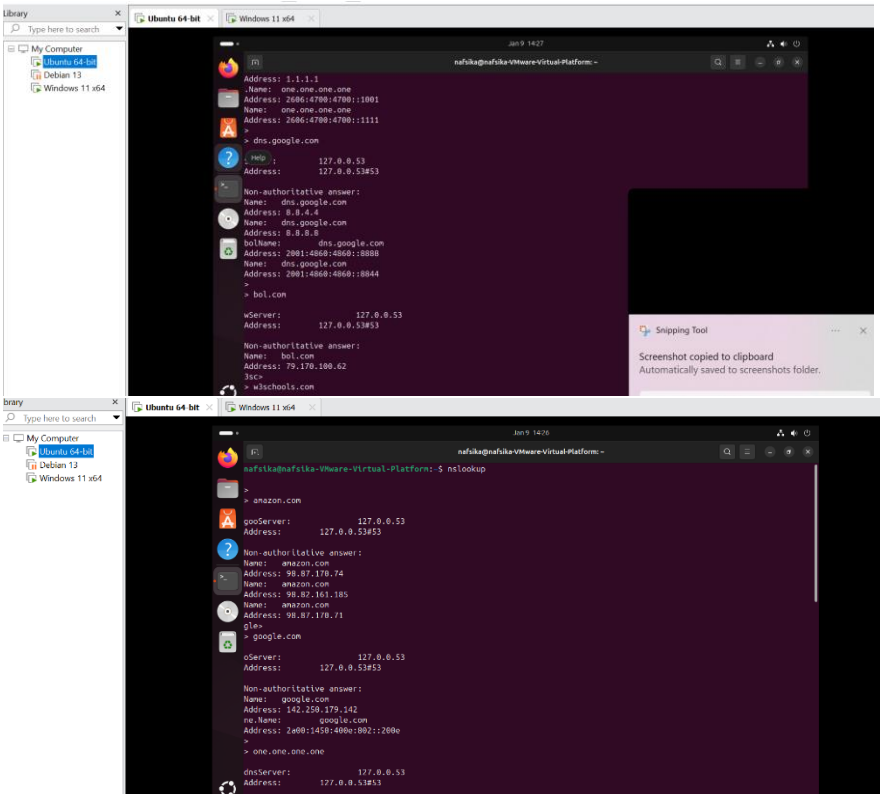


Screenshot remmina:

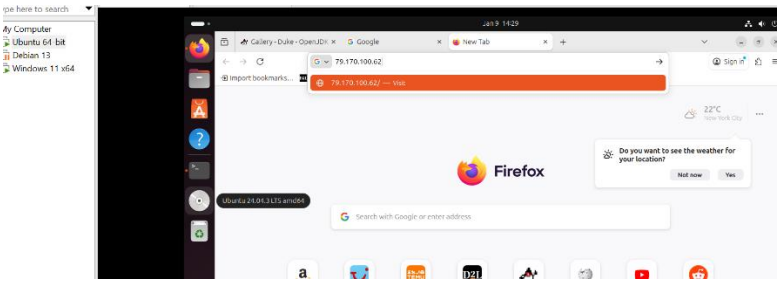


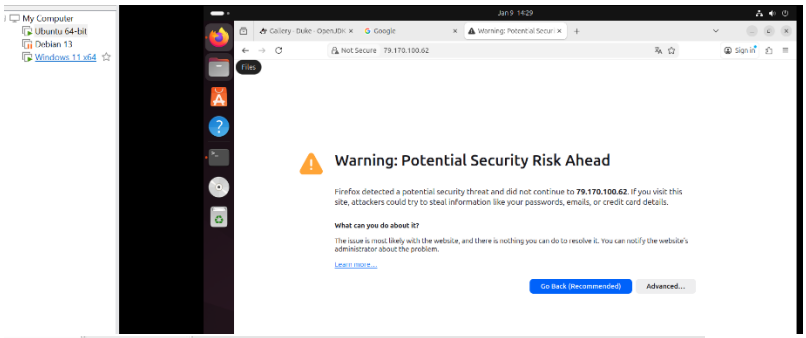
Assignment 6.2: IP addresses websites

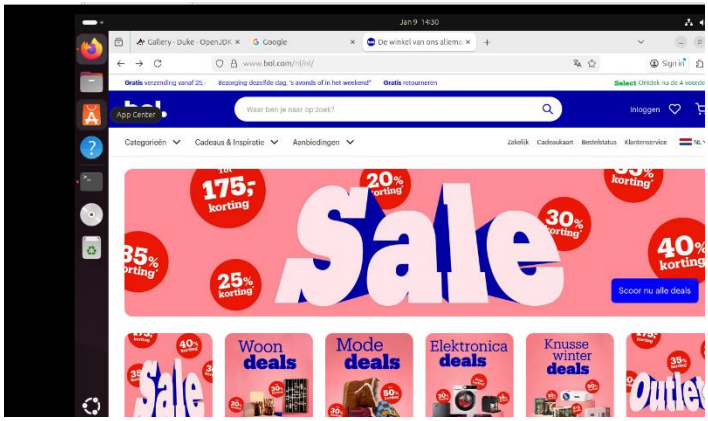
Relevant screenshots nslookup command:



Screenshot website visit via IP address:







Assignment 6.3: subnetting

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

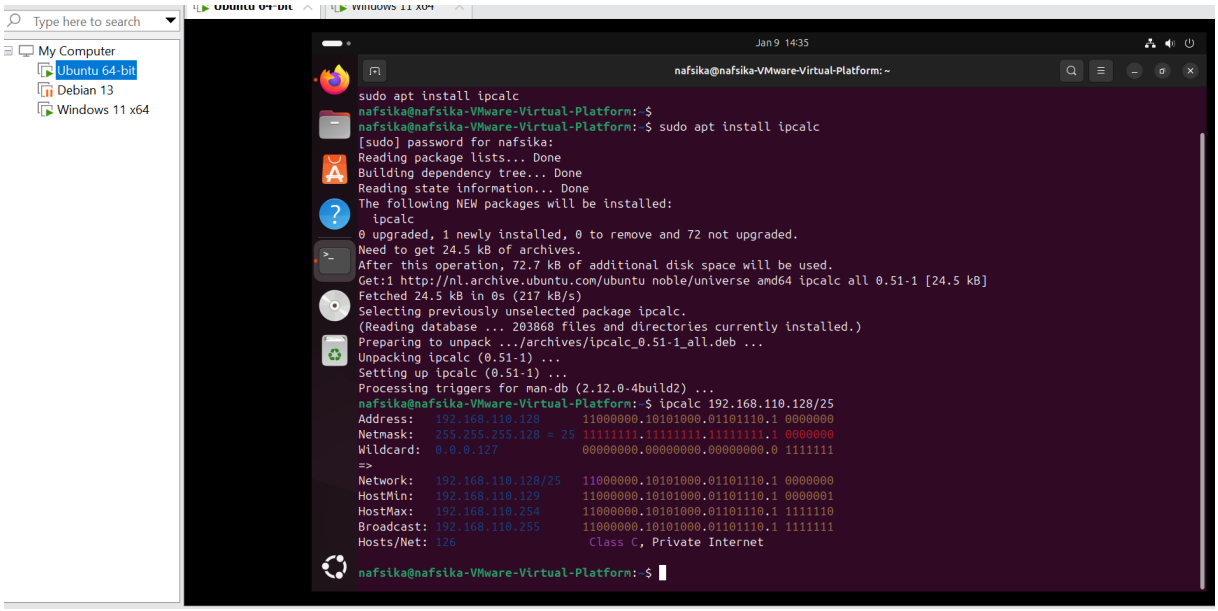
25 subnet means = 32 -25 = 7 bits

Number of IP addresses is 2 to the power of 7 = 128 Ips

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

192.168.110.129 – 192.168.110.254

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



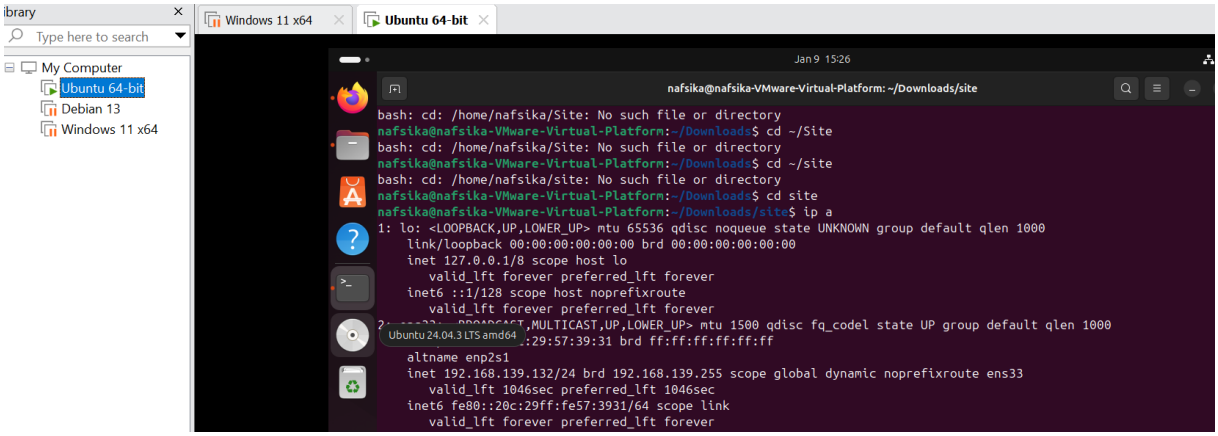
Explain the above calculation in your own words.

A /25 subnet uses 25 bits for the network part and 7 bits for hosts.

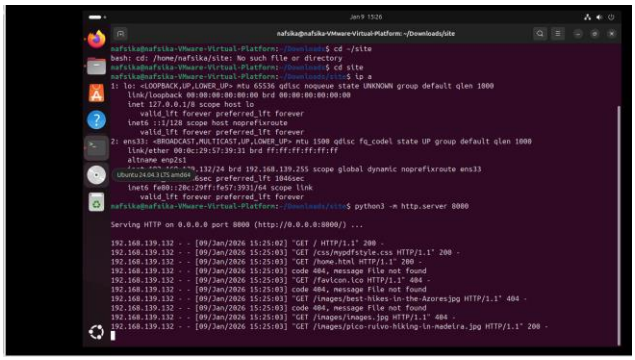
This results in 128 total IP addresses per subnet where the first address is reserved as the network address, and the last address is reserved as the broadcast address leaving 126 available IP addresse. Here the subnet starts at 192.168.110.128 and ends at 192.168.110.255.

Assignment 6.4: HTML

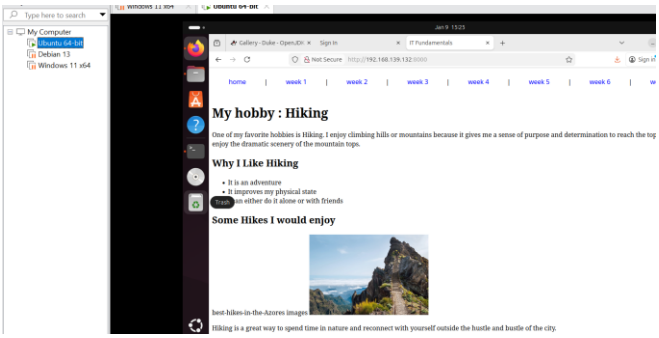
Screenshot IP address Ubuntu VM:



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

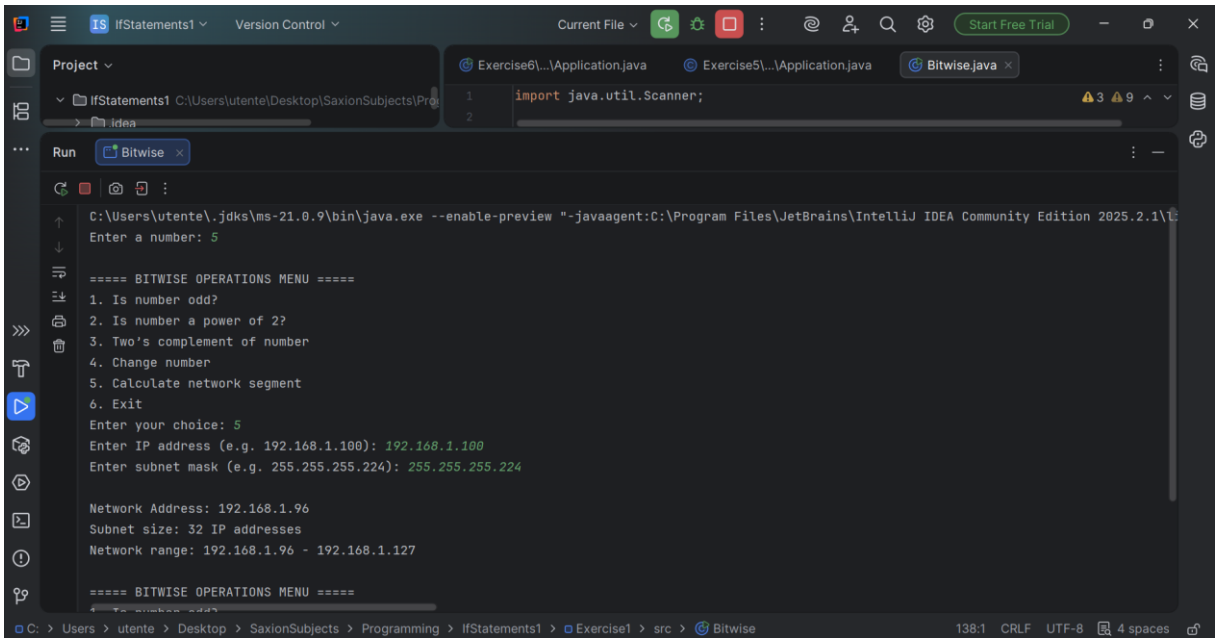
```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
-------------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000

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.

```
import java.util.Scanner;

public class Bitwise {

   public static void main(String[] args) {

      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter a number: ");
      int number = scanner.nextInt();
      scanner.nextLine(); // consume newline

      int choice = 0;

      while (choice != 6) {
         System.out.println("\n===== BITWISE OPERATIONS 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. Change number");
         System.out.println("5. Calculate network segment");
         System.out.println("6. Exit");
         System.out.print("Enter your choice: ");

         choice = scanner.nextInt();
         scanner.nextLine(); // consume newline

         switch (choice) {

            case 1:
               if (isOdd(number)) {
                  System.out.println(number + " is odd.");
               } else {
                  System.out.println(number + " is even.");
               }
               break;

            case 2:
               if (isPowerOfTwo(number)) {
                  System.out.println(number + " is a power of 2.");
               } else {
                  System.out.println(number + " is NOT a power of 2.");
               }
               break;
```

```
            case 3:
                System.out.println("Two's complement of " + number + " is: "
                        + twosComplement(number));
                break;

            case 4:
                System.out.print("Enter a new number: ");
                number = scanner.nextInt();
                scanner.nextLine();
                break;

            case 5:
                System.out.print("Enter IP address (e.g. 192.168.1.100): ");
                String ip = scanner.nextLine();

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

                calculateNetworkSegment(ip, subnet);
                break;

            case 6:
                System.out.println("Exiting program...");
                break;

            default:
                System.out.println("Invalid choice. Try again.");
        }
    }

    scanner.close();
}

// BITWISE METHODS

// Check if number is odd using bitwise AND
public static boolean isOdd(int num) {
    return (num & 1) == 1;
}

// Check if number is power of 2
public static boolean isPowerOfTwo(int num) {
    return num > 0 && (num & (num - 1)) == 0;
}

// Calculate two's complement
public static int twosComplement(int num) {
    return (~num) + 1;
```

```
    }

    //  NETWORK SEGMENT

    public static void calculateNetworkSegment(String ip, String subnet) {

        int ipInt = ipToInt(ip);
        int subnetInt = ipToInt(subnet);

        // Bitwise AND to get network address
        int networkInt = ipInt & subnetInt;

        // Calculate subnet size
        int hostBits = 32 - Integer.bitCount(subnetInt);
        int totalIPs = 1 << hostBits;

        int broadcast = networkInt + totalIPs - 1;

        System.out.println("\nNetwork Address: " + intToIp(networkInt));
        System.out.println("Subnet size: " + totalIPs + " IP addresses");
        System.out.println("Network range: " + intToIp(networkInt)
                + " - " + intToIp(broadcast));
    }

    // Convert dotted decimal IP to int
    public static int ipToInt(String ip) {
        String[] parts = ip.split("\\.");
        int result = 0;

        for (String part : parts) {
            result = (result << 8) | Integer.parseInt(part);
        }

        return result;
    }

    // Convert int back to dotted decimal IP
    public static String intToIp(int ip) {
        return ((ip >> 24) & 0xFF) + "." +
                ((ip >> 16) & 0xFF) + "." +
                ((ip >> 8) & 0xFF) + "." +
                (ip & 0xFF);
    }
}
```



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