Template Week 6 – Networking

Student number:

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:

Screenshot of Site directory contents:

Screenshot python3 webserver command:

Screenshot web browser visits your site

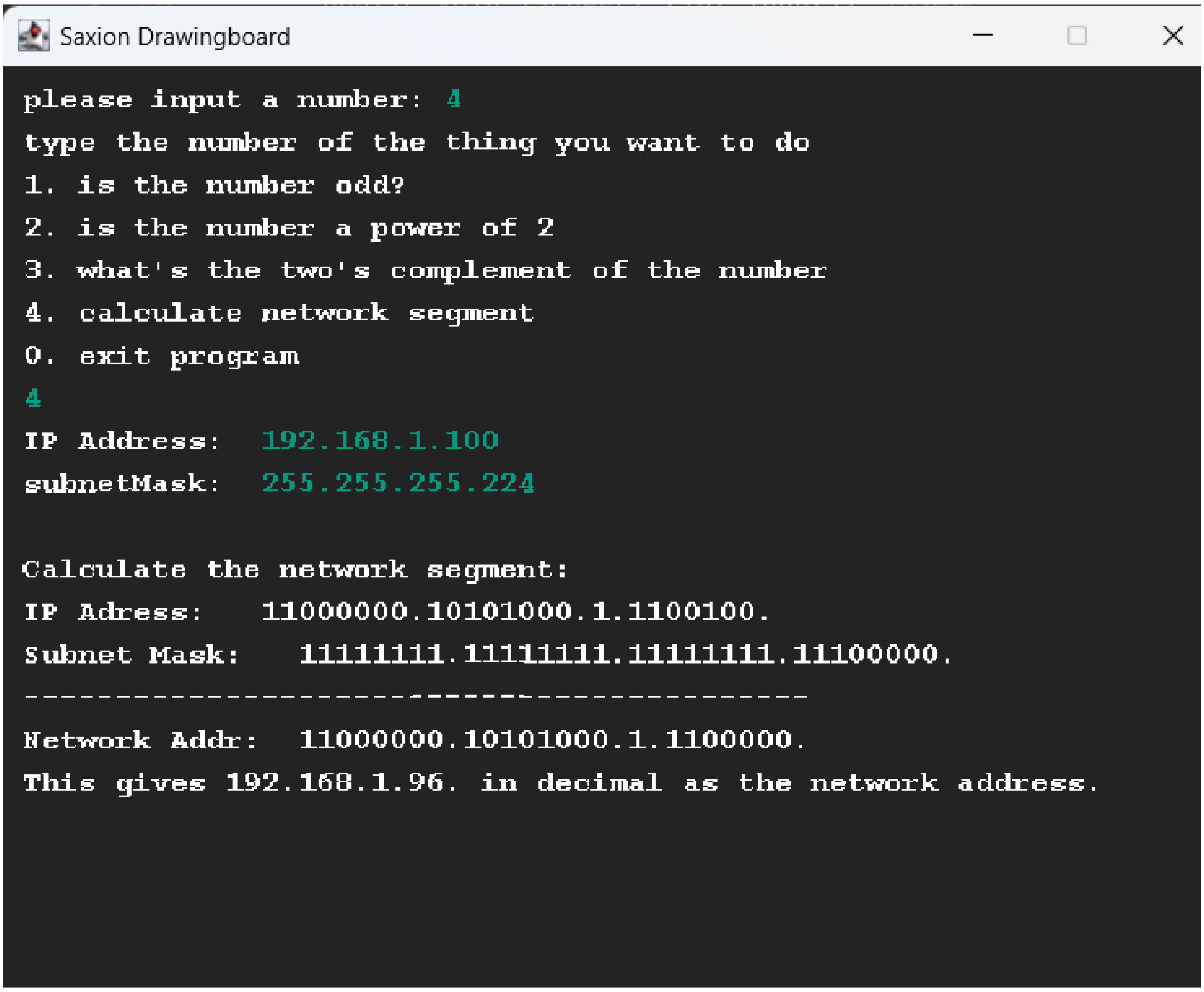
Bonus point assignment – week 6

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 (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```



```
please input a number: 4
type the number of the thing you want to do
1. is the number odd?
2. is the number a power of 2
3. what's the two's complement of the number
4. calculate network segment
0. exit program
4
IP Address:  192.168.1.100
subnetMask:  255.255.255.224

Calculate the network segment:
IP Adress:   11000000.10101000.1.1100100.
Subnet Mask:   11111111.11111111.11111111.11100000.
-------------------------------------------
Network Addr:  11000000.10101000.1.1100000.
This gives 192.168.1.96. in decimal as the network address.
```

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

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

import java.awt.*;

public class bitwise_program implements Runnable {

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

  public void run() {
    boolean running = true;
    boolean inputReady = true;
    int inputNumber = 0;

    while (running) {
      SaxionApp.clear();
```

```
            SaxionApp.print("please input a number: ");

            inputNumber = SaxionApp.readInt();



            SaxionApp.printLine("type the number of the thing you want to do");
            SaxionApp.printLine("1. is the number odd?");
            SaxionApp.printLine("2. is the number a power of 2");
            SaxionApp.printLine("3. what's the two's complement of the number");
            SaxionApp.printLine("4. calculate network segment");
            SaxionApp.printLine("0. exit program");
            int choice = SaxionApp.readInt();
            if(choice == 1){
                isOdd(inputNumber);
                SaxionApp.pause();

            } else if (choice == 2) {
                isPower2(inputNumber);
                SaxionApp.pause();

            } else if (choice == 3) {
                SaxionApp.printLine(twoComplement(inputNumber));
                SaxionApp.pause();

            } else if (choice == 4) {

                SaxionApp.print("IP Address: ");
                String IP_address = SaxionApp.readString();
                SaxionApp.print("subnetMask: ");
                String subnetMask = SaxionApp.readString();

                int[] network = calcNetworkSeg(IP_address, subnetMask);
                SaxionApp.printLine(" ");
                SaxionApp.print("This gives ");
                for(int i = 0; i< network.length; i++ ) {
                    SaxionApp.print(network[i]+".");
                }
                SaxionApp.print(" in decimal as the network address. ");

                SaxionApp.pause();

            } else if (choice == 0) {
                running = false;
            }
        }
    }
    public int[] calcNetworkSeg(String IP_address,String subnetMask){
```

```
String network_Addr = "1";

String [] IPSeperator = IP_address.split("[.]");
String [] SubnetSeperator  = subnetMask.split("[.]");

int[] IPint = new int[4];
int[] subInt= new int[4];
int[] netWorkInt = new int[4];

for(int i = 0; i< IPSeperator.length;  i++  ){
   int ipPart = Integer.parseInt(IPSeperator[i]);
   IPint[i] = ipPart;

}
for(int i = 0; i< SubnetSeperator.length;  i++  ){
   int subPart = Integer.parseInt(SubnetSeperator[i]);
   subInt[i] = subPart;
}

for(int i = 0; i< IPint.length;  i++  ) {
   netWorkInt[i] = IPint[i] & subInt[i];
}



SaxionApp.printLine("  ");
SaxionApp.printLine("Calculate  the network segment:   ");
SaxionApp.print("IP  Adress:   ");

for(int i = 0; i< IPint.length;  i++  ) {
   SaxionApp.print(Integer.toBinaryString(IPint[i])+".");
}
SaxionApp.printLine("  ");
SaxionApp.print("Subnet  Mask:   ");

for(int i = 0; i< subInt.length;  i++  ) {
   SaxionApp.print(Integer.toBinaryString(subInt[i])+".");
}
SaxionApp.printLine("  ");
SaxionApp.printLine("------------------------------------------- ");
SaxionApp.print("Network  Addr:  ");
for(int i = 0; i< netWorkInt.length;  i++  ) {
   SaxionApp.print(Integer.toBinaryString(netWorkInt[i])+".");
}


return netWorkInt;
```

```
    }

    public boolean isOdd (int input){
        boolean isOdd = false;
        int test1 = 1;
        int result1 = input & 1;
        if(result1 == 1){
            isOdd = true;
            SaxionApp.printLine("the number is odd");
        }else {
            SaxionApp.printLine("the number is even");
        }
        return isOdd;
    }

    public void isPower2 (int input){
        int test2 = input - 1;
        int check2 = input & test2;

        if (check2 == 0 && input !=1){
            SaxionApp.printLine("it's a power of two");
        }else{
            SaxionApp.printLine("it is NOT a power of two");
        }
    }

    public int twoComplement (int input){
        int complement = ~input + 1;
        return complement;
    }

}
```

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