Info I, Uebung 07

Aufgabe 1

1.)

```
public static boolean foo(int x) {
boolean b = true;
int t = 2;

while ((t * t <= x) && b) {
    b = ((x % t) != 0);
    t++;
}

boolean res = (b && (x != 1));
return res;
}
```

2.)
Ja, da t mit jedem Durchlauf um $+1$ erhöht und $t^2 <= x$ geprüft wird. Somit wird t nach n Durchläufen x schließlich übersteigen. Dazu gibt es noch den boolean von b, welcher innerhalb der Schleife durch b = ((x % t) != 0); verändert wird, wobei geprüft wird ob sich x ohne Rest durch t teilen lässt (wobei t++ bei jedem Durchlauf), falls ja $\rightarrow b = false \rightarrow$ Abbruch. Eine der beiden

Abbruchbedingungen wird immer zuerst eintreffen, bei negativen Zahlen wird die Schleife direkt abgebrochen.

3.)
Damit res = $true$ ist darf x != 1 sein. Dazu muss b = $true$ gelten. Damit b = $true$ ist hängt von der Schleife ab. Dabei kann b = $true$ nur dann erfüllt sein, wenn x **nicht** durch t teilbar ist, wobei t jede Zahl von 2 bis t_i durchläuft bis $t_i{}^2 >= x$ oder x durch t teilbar ist. Letzteres macht b zu $false$ d.h. damit $b = true$ darf x nicht durch die Zahlen 2 bis t_i teilbar sein bis $t_i{}^2 > x$. Somit bleibt am

Ende

eine Zahl z über mit $1 < z < t_i^2$, welche keinen Teiler von 2 bis t_i hat, was auf eine Primzahl schließen lässt. Für die Primzahl 2 (und 3) setzt die erste Abbruchbedingung t^2 <= x mit t = 2 ein, um zu verhindern dass b in der Schleife auf $false$ gesetzt wird, was wiederum res = $false$ setzen würde, obwohl 2 eine Primzahl ist. Demnach gibt die Methode foo $true$ aus, wenn x eine Primzahl ist, ansonsten $false$.

Aufgabe 2

1.)

```
public static void hornerUmkehrungRekursiv(int z, int q) {
        if (z == 0) {
         return;
        }

        hornerUmkehrungRekursiv(z / q, q);
        System.out.print(z % q);
}
```

2.)

```
hornerUmkehrungRekursiv(13, 2)
    res = 13 % 2 = 1
    hornerUmkehrungRekursiv(13/2, 2)
        res = 6 % 2 = 0
        hornerUmkehrungRekursiv(6/2, 2)
            res = 3 % 2 = 1
            hornerUmkehrungRekursiv(3/2, 2)
                res = 1 % 2 = 1
                hornerUmkehrungRekursiv(1/2, 2)
                    return
                output res = 1
            output res = 1
        output res = 0
    output res = 1
```