Aufgabe 1

Betrachten Sie die folgende Prozedur `countup`, die aus zwei ganzzahligen Eingabewerten `n` und `m` einen ganzzahligen Ausgabewert berechnet:

```
1  procedure countup(n, m : integer): integer
2  var x, y : integer;
3  begin
4    x := n;
5    y := 0;
6    while (y < m) do
7      x := x - 1;
8      y := y + 1;
9    end while
10   return x;
11 end
```

(a) Führen Sie `countup(3,2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen `n`, `m`, `x` und `y` zu Beginn der while-Schleife und den Rückgabewert der Prozedur an.

n	m	x	y	ausgeführter Code, der Änderung bewirkte
3	2	3	0	
3	2	2	1	`x := x - 1; y := y + 1;`
3	2	1	2	diese Schleife läuft nicht mehr durch `x := x - 1; y := y + 1;`

Rückgabewert: 1

```
3  public class CountUp {
4
5    public static int countup(int n, int m) {
6      int x = n;
7      int y = 0;
8      while (y < m) {
9        System.out.println(String.format("%s %s %s %s", n, m, x, y));
10       x = x - 1;
11       y = y + 1;
12     }
13     return x;
14   }
15
16   public static void main(String[] args) {
17     System.out.println(countup(3, 2));
18   }
19 }
```

(b) Gibt es Eingabewerte von `n` und `m`, für die die Prozedur `countup` nicht terminiert? Begründen Sie Ihre Antwort.

Nein. Mit jedem Schleifendurchlauf wird der Wert der Variablen `y` um eins hochgezählt, die Werte, die `y` annimmt, sind also mathematisch ausgedrückt streng monoton steigend. `y` nähert sich `m` an, bis `y` nicht mehr kleiner ist als `m` und die Prozedur terminiert. An diesem Sachverhält ändern auch sehr große Zahlen, die über die Variable `m` der Prozedur übergeben werden, nichts.

(c) Geben Sie die asymptotische worst-case Laufzeit der Prozedur `countup` in der Θ-Notation in Abhängigkeit von den Eingabewerten `n` und/oder `m` an. Begründen Sie Ihre Antwort.

> Die Laufzeit der Prozedur ist immer $\Theta(m)$. Die Laufzeit hängt nur von `m` ab. Es kann nicht zwischen best-, average and worst-case unterschieden werden.

(d) Betrachten Sie nun die folgende Prozedur `countdown`, die aus zwei ganzzahligen Eingabewerten `n` und `m` einen ganzzahligen Ausgabewert berechnet:

```
procedure countdown(n, m : integer) : integer
var x, y : integer;
begin
  x := n;
  y := 0;
  while (n > 0) do
    if (y < m) then
      x := x - 1;
      y := y + 1;
    else
      y := 0;
      n := n / 2; /* Ganzzahldivision */
    end if
  end while
  return x;
end
```

```
public class CountDown {

  public static int countdown(int n, int m) {
    int x = n;
    int y = 0;
    while (n > 0) {
      System.out.println(String.format("%s %s %s %s", n, m, x, y));
      if (y < m) {
        x = x - 1;
        y = y + 1;
      } else {
        y = 0;
        n = n / 2; /* Ganzzahldivision */
      }
    }
    return x;
  }

  public static void main(String[] args) {
    System.out.println(countdown(3, 2));
  }

}
```

Führen Sie `countdown(3, 2)` aus. Geben Sie für jeden Schleifendurchlauf jeweils den Wert der Variablen `n`, `m`, `x` und `y` zu Beginn der `while`-Schleife und den Rückgabewert der Prozedur an.

n	m	x	y	ausgeführter Code, der Änderung bewirkte
3	2	3	0	
3	2	2	1	`x := x - 1; y := y + 1;`
3	2	1	2	`x := x - 1; y := y + 1;`
1	2	1	0	`y := 0; n := n / 2;`
1	2	0	1	`x := x - 1; y := y + 1;`
1	2	-1	2	`x := x - 1; y := y + 1;`
0	2	-1	0	Wert am Ende der While-Schleife, nicht mehr Beginn der while-Schleife

Rückgabewert: -1

(e) Gibt es Eingabewerte von `n` und `m`, für die die Prozedur `countdown` nicht terminiert?

Begründen Sie Ihre Antwort.

Nein.

`n <= 0` terminiert sofort

`m <= 0` Falsch-Block der Wenn-Dann-Bedingung erniedrigt `n` `n := n / 2;` bis `0` erreicht ist. Dann terminiert die Prozedur.

`m > 0` 1. Wahr-Block der Wenn-Dann-Bedingung erhöht `y` streng monoton bis `y >= m`. 2. Falsch-Block der Wenn-Dann-Bedingung halbiert `n` bis `0`. 1. und 2. solange bis n = 0

(f) Geben Sie die asymptotische Laufzeit der Prozedur `countdown` in der Θ-Notation in Abhängigkeit von den Eingabewerten `n` und/oder `m` an unter der Annahme, dass `m >= 0` und `n > 0`. Begründen Sie Ihre Antwort.

Anzahl der while Schleife:

m + 1:

m oft: bis y < m +1 Halbierung von n und y auf 0 setzen

wegen dem n/2 ist die Laufzeit logarithmisch, ähnlich wie worst case bei der Binären Suche.

n	m	x	y	ausgeführter Code, der Änderung bewirkte
16	3	16	0	
16	3	15	1	
16	3	14	2	
16	3	13	3	
8	3	13	0	`y := 0; n := n / 2;`
8	3	12	1	
8	3	11	2	
8	3	10	3	
4	3	10	0	`y := 0; n := n / 2;`
4	3	9	1	
4	3	8	2	
4	3	7	3	
2	3	7	0	`y := 0; n := n / 2;`
2	3	6	1	
2	3	5	2	
2	3	4	3	
1	3	4	0	`y := 0; n := n / 2;`
1	3	3	1	
1	3	2	2	
1	3	1	3	

$\Theta((m+1)\log_2 n)$

Wegkürzen der Konstanten

$\Theta(m \log n)$