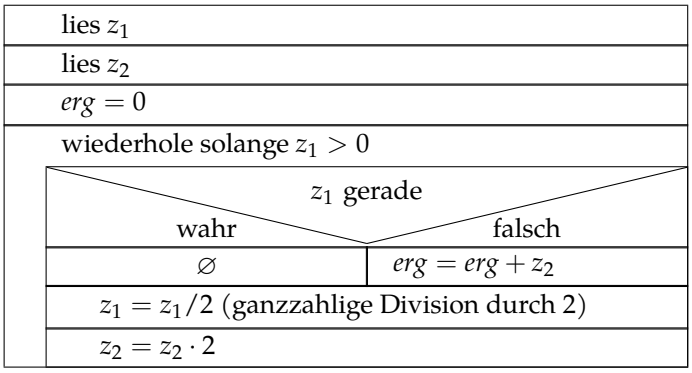
Abitur 2015 IV

Auf dem ägyptischen *Papyrus Rhind*, der etwa auf das Jahr 1550 v. Chr. datiert wird, ist eine Möglichkeit zur Multiplikation zweier natürlicher Zahlen z_1 und z_2 beschrieben. Als Struktogramm lässt sich dieser Algorithmus folgendermaßen darstellen:



Das berechnete Produkt steht nach Abarbeitung des Algorithmus in der Variablen *erg*.

(a) Berechnen Sie mithilfe der beschriebenen ägyptischen Multiplikation schrittweise das Produkt aus $z_1 = 13$ und $z_2 = 5$

z_1	z_2	*erg*
13	5	5
6	10	-
3	20	25
1	40	65
		65

(b) Nennen Sie die wesentliche Idee des Speichermodells eines Rechners, der nach dem von-Neumann-Prinzip aufgebaut ist. Geben Sie einen Vor- und einen Nachteil dieses Speichermodells an.

Wesentliche Idee des Speichermodells eines Rechners, der nach dem von-Neumann-Prinzip gebaut ist: Programme und Daten sind im selben Speicher, wobei der Hauptspeicher aus Zellen gleicher Größe besteht.

Vorteil:

Streng sequentieller Ablauf von Befehlen ist ein Vorteil, weil zu jedem Zeitpunkt klar ist, welcher Schritt durchgeführt wird.

Nachteil:

Der von-Neumann-Flaschenhals, weil alle Daten über denselben Bus weitergeleitet werden müssen und der Ablauf deshalb eine gewisse Zeit benötigt.

(c) Bestätigen Sie anhand zweier Beispiele, dass mithilfe des folgenden Programmausschnitts entschieden werden kann, ob Speicherzelle 101 eine gerade oder ungerade Zahl enthält.

```
LOAD 101
SHRI 1
SHLI 1
SUB 101
```

	gerade Zahl	ungerade Zahl
`LOAD 101`	8 (0b1000)	9 (0b1001)
`SHRI 1`	4 (0b0100)	4 (0b0100)
`SHLI 1`	8 (0b1000)	8 (0b1000)
`SUB 101`	0	-1

(d) Schreiben Sie ein Programm für die angegebene Registermaschine, das den Algorithmus des *Papyrus Rhind* umsetzt. Gehen Sie dabei davon aus, dass die beiden positiven ganzzahligen Faktoren z_1 und z_2 bereits in den Speicherzellen 101 und 102 stehen und dass alle weiteren nicht vom Programm belegten Speicherzellen mit dem Wert 0 vorbelegt sind.

```
Start:  LOADI 13
 STORE 101
 LOADI 5
 STORE 102

Wdh: LOAD 101
 JMPZ end

Bed: SHRI 1
 SHLI 1
 SUB 101
 CMPI 0
 JMPZ true

false: LOAD erg
 ADD 102
 STORE erg

true: LOAD 101
 DIVI 2
 STORE 101
 LOAD 102
 MULI 2
 STORE 102
 JMP Wdh

end:  LOAD erg
 HOLD

erg: WORD 0
```

Java (iterativ)

```
  public static int multipliziereIterativ(int z1, int z2) {
    int ergebnis = 0;
    while (z1 > 0) {
      if (z1 % 2 == 1) {
```

```
10          ergebnis = ergebnis + z2;
11        }
12        z1 = z1 / 2;
13        z2 = z2 * 2;
14      }
15      return ergebnis;
16    }
```

Java (rekursiv)

```
18    public static int multipliziereRekursiv(int z1, int z2) {
19      if (z1 == 1)
20        return z2;
21      int z1Alt = z1;
22      int z2Alt = z2;
23
24      z1 = z1 / 2;
25      z2 = z2 * 2;
26
27      if (z1Alt % 2 == 1)
28        return z2Alt + multipliziereRekursiv(z1, z2);
29      else
30        return multipliziereRekursiv(z1, z2);
31    }
```