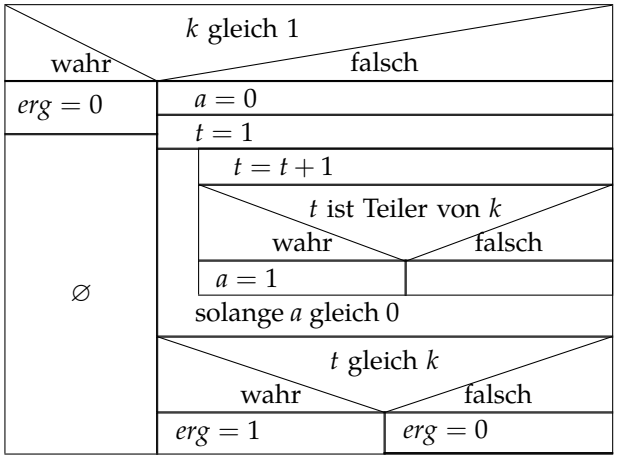
Abitur 2017 IV

In dem folgenden Struktogramm wird ein Algorithmus dargestellt, der erkennt, ob eine natürliche Zahl k eine Primzahl ist. In diesem Fall wir in die Speicherzelle *erg* die Zahl 1 abgelegt, sonst 0.



(a) Stellen Sie die Veränderung der Variablenwerte bei Ablauf dieses Algorithmus jeweils für die Startwerte $k = 5$ und $k = 15$ durch zwei Speicherbelegungstabellen wie nachfolgend gezeigt dar.

Anweisung	k	a	t	erg
	5			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$	5		2	

$k = 5$

Anweisung	k	a	t	erg
	5			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$			2	
$t = t + 1$			3	
$t = t + 1$			4	
$t = t + 1$			5	
$a = 1$		1		
$erg = 1$				1

$k = 15$

Anweisung	k	a	t	erg
	15			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$			2	
$t = t + 1$			3	
$t = t + 1$			4	
$t = t + 1$		1	5	
$a = 1$		1		
$erg = 0$				0

Im Folgenden soll ein Programm für diese Maschine erstellt werden, das den dargestellten Algorithmus umsetzt. Der Wert von k soll in Speicherzelle 101, der von a in 102, der von t in 103 und der von erg in 104 gespeichert werden.

(b) Betrachten Sie die folgende kurze Sequenz; xx steht dabei für ein geeignetes Sprungziel.

```
1  LOAD 101
2  MOD 103
3  JMPP xx
4  LOADI 1
5  STORE 102
```

Geben Sie an, welcher Teil des Algorithmus damit umgesetzt wird.

> Die Bedingung „t ist Teiler von k“.

(c) Setzen Sie unter Verwendung der Sequenz aus Teilaufgabe 2b den gesamten Algorithmus in eine Programm für die gegebene Registermaschine um.

Assembler

```
1  # k: 101
2  # a: 102
3  # t: 103
4  # erg: 104
5
6  # IF k = 1 THEN
7                  LOADI 101
8                  CMPI 1
9                  JMPZ ist_nicht_prim
10
11 # a := 1;
12                 LOADI 0
13                 STORE 102
14
15 # t := 1;
16                 LOADI 1
17                 STORE 103
18
19 # t := t + 1;
20 erhoehe_t:     LOAD 103
21                 ADDI 1
22                 STORE 103
23
24 # IF (k % t) = 0 THEN
25                 LOAD 101
26                 MOD 103
27                 JMPP solange_bed
28
29 # a := 1;
30                 LOADI 1
31                 STORE 102
32
33 # UNTIL a = 0;
34 solange_bed:   LOAD 102
35                 JMPZ erhoehe_t
36
37 # IF t = k THEN;
38                 LOAD 103
39                 CMP 101
```

```
                JMPZ ist_prim

# erg := 0;
ist_nicht_prim: LOADI 0
                STORE 104
                JMP ende

# erg := 1;
ist_prim:       LOADI 1
                STORE 104

ende:           HOLD
```

Minisprache

```
PROGRAM primzahl;
VAR k, a, t, erg;

BEGIN
  k := 5;
  IF k = 1 THEN
    erg := 0;
  ELSE
    a := 0;
    t := 1;
    REPEAT
      t := t + 1;
      IF (k % t) = 0 THEN
        a := 1;
      END
    UNTIL a = 0;

    IF t = k THEN
      erg := 1;
    ELSE
      erg := 0;
    END
  END;
END primzahl.
```

Java

```
  public static boolean istPrimzahl(int k) {
    if (k == 1)
      return false;
    int a = 0;
    int t = 1;

    do {
      t++;
      if (k % t == 0) {
        a = 1;
      }
    } while (a == 0);

    if (t == k) {
      return true;
    } else {
```

```
21          return false;
22        }
23      }
```