

Übungsblatt 4

Adrian Schollmeyer

Aufgabe 1

(a)

Listing 1: GZahlMain.java

```
1 package me.adrian;
2
3 public class GZahlMain
4 {
5     private static final long MOD_N = 100000000001;
6
7     public static long bPotSum(short n)
8     {
9         long buf = 0l;
10        for (short i = 1; i <= n; i++) {
11            buf += modPow(i, i);
12            buf %= MOD_N;
13        }
14        return buf;
15    }
16
17    public static long modPow(int b, int e)
18    {
19        long buf = 1l;
20        if (e == 1)
21            return b;
22
23        for (int i = 0; i < b; i++) {
24            buf = (buf * b) % MOD_N;
25            Thread.yield();
26        }
27        return buf;
28    }
29
30    public static void main(String[] args)
```

```

31   {
32       System.out.printf("%010d", bPotSum(Short.parseShort(args[0])));
33   }
34 }
```

(b)

Die letzten zehn Ziffern der Summe für $n = 1337$ sind 2525972725.

Aufgabe 2

$$Z = \llbracket \alpha \rrbracket (Z_0) \quad (1)$$

$$= \llbracket \alpha \rrbracket (2, \perp) \quad (2)$$

$$= \llbracket q := 0; \text{while } \beta \rrbracket (2, \perp) \quad (3)$$

$$= \llbracket \text{while } \beta \rrbracket (2, \perp)_{\langle q \leftarrow 0 \rangle} \quad (4)$$

$$= \llbracket \text{while } \beta \rrbracket (2, 0) \quad (5)$$

$$= \begin{cases} Z & , \text{ falls } Z(B) = \text{false} \\ \llbracket \text{while } \beta \text{ do } \alpha' \text{ od } \rrbracket (\llbracket \alpha' \rrbracket (Z)) & , \text{ sonst} \end{cases} \quad (6)$$

$$= \begin{cases} (2, 0) & , \text{ falls } Z(n > 0) = (2 > 0) = \text{false} \\ \llbracket \text{while } \beta \rrbracket (\llbracket q := q + n + n - 1; n := n - 1 \rrbracket (Z)) & , \text{ sonst} \end{cases} \quad (7)$$

$$= \llbracket \text{while } \beta \rrbracket (\llbracket q := q + n + n - 1; n := n - 1 \rrbracket (2, 0)) \quad (8)$$

$$= \llbracket \text{while } \beta \rrbracket (\llbracket n := n - 1 \rrbracket (\llbracket q := q + n + n - 1 \rrbracket (2, 0))) \quad (9)$$

$$= \llbracket \text{while } \beta \rrbracket (\llbracket n := n - 1 \rrbracket (2, 3)) \quad (10)$$

$$= \llbracket \text{while } \beta \rrbracket (1, 3) \quad (11)$$

$$= \begin{cases} (2, 3) & , \text{ falls } Z(n > 0) = (1, 3) = \text{false} \\ \llbracket \text{while } \beta \rrbracket (\llbracket q := q + n + n - 1; n := n - 1 \rrbracket (Z)) & , \text{ sonst} \end{cases} \quad (12)$$

$$= \llbracket \text{while } \beta \rrbracket (\llbracket n := n - 1 \rrbracket (\llbracket q := q + n + n - 1 \rrbracket (1, 3))) \quad (13)$$

$$= \llbracket \text{while } \beta \rrbracket (\llbracket n := n - 1 \rrbracket (1, 4)) \quad (14)$$

$$= \llbracket \text{while } \beta \rrbracket (0, 4) \quad (15)$$

$$= \begin{cases} (0, 4) & , \text{ falls } Z(n > 0) = (0 > 0) = \text{false} \\ \llbracket \text{while } \beta \rrbracket (\llbracket q := q + n + n - 1; n := n - 1 \rrbracket (Z)) & , \text{ sonst} \end{cases} \quad (16)$$

$$= (0, 4) \quad (17)$$

Aufgabe 3

$$\llbracket \gamma; \text{while } B \text{ do } \alpha \text{ od} \rrbracket (Z) = \begin{cases} Z & , \text{ falls } Z(B) = \text{false} \\ \llbracket \text{while } B \text{ do } \alpha \text{ od} \rrbracket (\llbracket \delta \rrbracket (Z)) & , \text{ sonst} \end{cases} (\llbracket \gamma \rrbracket (Z))$$

Aufgabe 4

(a)

$$add(4, 2, 6) \tag{18}$$

$$\implies add(4, y', z'), suc(y', 2), suc(z', 6) \tag{19}$$

$$\implies y' = 1 \text{ (Fakt)} \tag{20}$$

$$\implies z' = 5 \text{ (Fakt)} \tag{21}$$

$$\implies add(4, 1, 5) \tag{22}$$

$$\implies add(4, y'', z''), suc(y'', 1), suc(z'', 5) \tag{23}$$

$$\implies y'' = 0, z'' = 0 \text{ (Fakten)} \tag{24}$$

$$\implies add(4, 0, 4) \text{ wahr nach Regel 1} \tag{25}$$

$$\implies add(4, 2, 6) \text{ wahr } \square \tag{26}$$

(b)

- $add(y, z, x) \implies minus(x, y, z)$
- $\implies mult(x, 1, x)$
- $\implies mult(x, 0, 0)$
- $mult(x, y, z) \wedge suc(y, v) \wedge add(x, z, w) \implies mult(x, v, w)$
- $mult(y, z, x) \implies div(x, z, y)$
- $mult(y, z, x) \implies div(x, y, z)$

Aufgabe 5

Listing 2: Maze.java (Teil 1)

```
1 package me.adrian;
2
3 public class Maze
4 {
5
6     static int[][] maze = { { 2, 6, 12, 12, 10, 6, 10, 4, 12, 10 }, { 5, 9,
7         6, 8, 5, 9, 5, 12, 10, 3 },
8         { 6, 8, 7, 12, 12, 14, 8, 6, 9, 3 }, { 7, 10, 3, 2, 6, 9, 6, 9,
9         4, 11 },
10        { 3, 5, 9, 3, 5, 12, 9, 4, 14, 11 }, { 3, 6, 10, 7, 12, 12, 14,
11         14, 9, 3 },
12        { 3, 1, 7, 9, 6, 10, 1, 3, 4, 9 }, { 5, 10, 5, 8, 3, 5, 10, 5,
13         12, 10 },
14        { 2, 5, 12, 12, 9, 2, 5, 12, 12, 11 }, { 5, 12, 12, 12, 12, 13,
15         12, 12, 12, 9 } };
16
17     static int h = maze.length; // height of the maze
18     static int w = maze[0].length; // width of the maze
19
20     static int s_x = 0; // start at this x-coordinate
21     static int s_y = 0; // start at this y-coordinate
22     static int t_x = w - 1; // destination is at this x-coordinate
23     static int t_y = h - 1; // destination is at this y-coordinate
24
25     static int lastX, lastY; // Just some buffering that we don't follow the
26         path backwards
```

Listing 3: Maze.java (Teil 2)

```
1     public static boolean search()
2     {
3         setWaypoint(s_x, s_y);
4         if (check(s_x, s_y, 'N')) {
5             if (s_x == t_x && s_y - 1 == t_y) {
6                 setWaypoint(s_x, s_y);
7                 return true;
8             }
9
10            lastX = s_x;
11            lastY = s_y;
12            if (searchRec(s_x, s_y - 1)) {
13                System.out.println("Gotcha!");
14                setWaypoint(t_x, t_y);
15                return true;
16            }
17        }
```

```
18     if (check(s_x, s_y, 'S')) {
19         if (s_x == t_x && s_y + 1 == t_y) {
20             setWaypoint(s_x, s_y);
21             return true;
22         }
23
24         lastX = s_x;
25         lastY = s_y;
26         if (searchRec(s_x, s_y + 1)) {
27             System.out.println("Gotcha!");
28             setWaypoint(t_x, t_y);
29             return true;
30         }
31     }
32
33     if (check(s_x, s_y, 'E')) {
34         if (s_x + 1 == t_x && s_y == t_y) {
35             setWaypoint(s_x, s_y);
36             return true;
37         }
38
39         lastX = s_x;
40         lastY = s_y;
41         if (searchRec(s_x + 1, s_y)) {
42             System.out.println("Gotcha!");
43             setWaypoint(t_x, t_y);
44             return true;
45         }
46     }
47     if (check(s_x, s_y, 'W')) {
48         if (s_x - 1 == t_x && s_y == t_y) {
49             setWaypoint(s_x, s_y);
50             return true;
51         }
52
53         lastX = s_x;
54         lastY = s_y;
55         if (searchRec(s_x - 1, s_y)) {
56             System.out.println("Gotcha!");
57             setWaypoint(t_x, t_y);
58             return true;
59         }
60     }
61     return false;
62 }
63
64 /// helper method for a recursive solution
65 public static boolean searchRec(int x, int y)
66 {
```

```
67     setWaypoint(x, y);
68
69     if (check(x, y, 'N') && !(lastX == x && lastY == y - 1)) {
70         if (x == t_x && y - 1 == t_y) {
71             return true;
72         }
73
74         lastX = x;
75         lastY = y;
76         if (searchRec(x, y - 1))
77             return true;
78     }
79     if (check(x, y, 'S') && !(lastX == x && lastY == y + 1)) {
80         if (x == t_x && y + 1 == t_y) {
81             return true;
82         }
83
84         lastX = x;
85         lastY = y;
86         if (searchRec(x, y + 1))
87             return true;
88     }
89     if (check(x, y, 'E') && !(lastX == x + 1 && lastY == y)) {
90         if (x + 1 == t_x && y == t_y) {
91             return true;
92         }
93
94         lastX = x;
95         lastY = y;
96         if (searchRec(x + 1, y))
97             return true;
98     }
99     if (check(x, y, 'W') && !(lastX == x - 1 && lastY == y)) {
100        if (x - 1 == t_x && y == t_y) {
101            return true;
102        }
103
104        lastX = x;
105        lastY = y;
106        if (searchRec(x - 1, y))
107            return true;
108    }
109
110    unsetWaypoint(x, y);
111    return false;
112 }
```