Übungsblatt 4

Adrian Schollmeyer

Aufgabe 1

(a)

Listing 1: GZahlMain.java

```
package me.adrian;

public class GZahlMain
{
    private static final long MOD_N = 100000000001;

    public static long bPotSum(short n)
    {
        long buf = 0l;
        for (short i = 1; i <= n; i++) {
            buf += modPow(i, i);
            buf %= MOD_N;
        }
        return buf;
    }

    public static long modPow(int b, int e)
    {
        long buf = 1l;
        if (e == 1)
            return b;

        for (int i = 0; i < b; i++) {
            buf = (buf * b) % MOD_N;
            Thread.yield();
        }
        return buf;
    }

    public static void main(String[] args)
```

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

(b)

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

Aufgabe 2

$$
\begin{aligned}
Z &= [\![\alpha]\!]\,(Z_0) && (1)\\
&= [\![\alpha]\!]\,(2,\ \bot) && (2)\\
&= [\![q := 0;\ \text{while } \beta]\!]\,(2,\ \bot) && (3)\\
&= [\![\text{while } \beta]\!]\,(2,\ \bot)_{\langle q \leftarrow 0\rangle} && (4)\\
&= [\![\text{while } \beta]\!]\,(2, 0) && (5)\\
&= \begin{cases} Z & \text{, falls } Z(B) = \mathit{false} \\ [\![\text{while } \beta \text{ do } \alpha' \text{ od }]\!]\,([\![\alpha']\!]\,(Z)) & \text{, sonst} \end{cases} && (6)\\
&= \begin{cases} (2, 0) & \text{, falls } Z(n > 0) = (2 > 0) = \mathit{false} \\ [\![\text{while } \beta]\!]\,([\![q := q + n + n - 1; n := n - 1]\!]\,(Z)) & \text{, sonst} \end{cases} && (7)\\
&= [\![\text{while } \beta]\!]\,([\![q := q + n + n - 1; n := n - 1]\!]\,(2, 0)) && (8)\\
&= [\![\text{while } \beta]\!]\,([\![n := n - 1]\!]\,([\![q := q + n + n - 1]\!]\,(2, 0))) && (9)\\
&= [\![\text{while } \beta]\!]\,([\![n := n - 1]\!]\,(2, 3)) && (10)\\
&= [\![\text{while } \beta]\!]\,(1, 3) && (11)\\
&= \begin{cases} (2, 3) & \text{, falls } Z(n > 0) = (1, 3) = \mathit{false} \\ [\![\text{while } \beta]\!]\,([\![q := q + n + n - 1; n := n - 1]\!]\,(Z)) & \text{, sonst} \end{cases} && (12)\\
&= [\![\text{while } \beta]\!]\,([\![n := n - 1]\!]\,([\![q := q + n + n - 1]\!]\,(1, 3))) && (13)\\
&= [\![\text{while } \beta]\!]\,([\![n := n - 1]\!]\,(1, 4)) && (14)\\
&= [\![\text{while } \beta]\!]\,(0, 4) && (15)\\
&= \begin{cases} (0, 4) & \text{, falls } Z(n > 0) = (0 > 0) = \mathit{false} \\ [\![\text{while } \beta]\!]\,([\![q := q + n + n - 1; n := n - 1]\!]\,(Z)) & \text{, sonst} \end{cases} && (16)\\
&= (0, 4) && (17)
\end{aligned}
$$

Aufgabe 3

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

Aufgabe 4

(a)

$$\begin{aligned}
& \mathit{add}(4,2,6) && (18)\\
& \implies \mathit{add}(4,y',z'), \mathit{suc}(y',2), \mathit{suc}(z',6) && (19)\\
& \implies y' = 1(\text{Fakt}) && (20)\\
& \implies z' = 5(\text{Fakt}) && (21)\\
& \implies \mathit{add}(4,1,5) && (22)\\
& \implies \mathit{add}(4,y'',z''), \mathit{suc}(y'',1), \mathit{suc}(z'',5) && (23)\\
& \implies y'' = 0, z'' = 0(\text{Fakten}) && (24)\\
& \implies \mathit{add}(4,0,4) \text{ wahr nach Regel 1} && (25)\\
& \implies \mathit{add}(4,2,6) \text{ wahr} \quad \square && (26)
\end{aligned}$$

(b)

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

Aufgabe 5

Listing 2: Maze.java (Teil 1)

```
package me.adrian;

public class Maze
{

    static int[][] maze = { { 2, 6, 12, 12, 10, 6, 10, 4, 12, 10 }, { 5, 9, 6, 8, 5, 9, 5, 12, 10, 3 },
            { 6, 8, 7, 12, 12, 14, 8, 6, 9, 3 }, { 7, 10, 3, 2, 6, 9, 6, 9, 4, 11 },
            { 3, 5, 9, 3, 5, 12, 9, 4, 14, 11 }, { 3, 6, 10, 7, 12, 12, 14, 14, 9, 3 },
            { 3, 1, 7, 9, 6, 10, 1, 3, 4, 9 }, { 5, 10, 5, 8, 3, 5, 10, 5, 12, 10 },
            { 2, 5, 12, 12, 9, 2, 5, 12, 12, 11 }, { 5, 12, 12, 12, 12, 13, 12, 12, 12, 9 } };

    static int h = maze.length; // height of the maze
    static int w = maze[0].length; // width of the maze

    static int s_x = 0; // start at this x-coordinate
    static int s_y = 0; // start at this y-coordinate
    static int t_x = w - 1; // destination is at this x-coordinate
    static int t_y = h - 1; // destination is at this y-coordinate

    static int lastX, lastY; // Just some buffering that we don't follow the path backwards
```

Listing 3: Maze.java (Teil 2)

```
    public static boolean search()
    {
        setWaypoint(s_x, s_y);
        if (check(s_x, s_y, 'N')) {
            if (s_x == t_x && s_y - 1 == t_y) {
                setWaypoint(s_x, s_y);
                return true;
            }

            lastX = s_x;
            lastY = s_y;
            if (searchRec(s_x, s_y - 1)) {
                System.out.println("Gotcha!");
                setWaypoint(t_x, t_y);
                return true;
            }
        }
```

```
        if (check(s_x, s_y, 'S')) {
            if (s_x == t_x && s_y + 1 == t_y) {
                setWaypoint(s_x, s_y);
                return true;
            }

            lastX = s_x;
            lastY = s_y;
            if (searchRec(s_x, s_y + 1)) {
                System.out.println("Gotcha!");
                setWaypoint(t_x, t_y);
                return true;
            }

        }
        if (check(s_x, s_y, 'E')) {
            if (s_x + 1 == t_x && s_y == t_y) {
                setWaypoint(s_x, s_y);
                return true;
            }

            lastX = s_x;
            lastY = s_y;
            if (searchRec(s_x + 1, s_y)) {
                System.out.println("Gotcha!");
                setWaypoint(t_x, t_y);
                return true;
            }
        }
        if (check(s_x, s_y, 'W')) {
            if (s_x - 1 == t_x && s_y == t_y) {
                setWaypoint(s_x, s_y);
                return true;
            }

            lastX = s_x;
            lastY = s_y;
            if (searchRec(s_x - 1, s_y)) {
                System.out.println("Gotcha!");
                setWaypoint(t_x, t_y);
                return true;
            }
        }
        return false;
    }

    /// helper method for a recursive solution
    public static boolean searchRec(int x, int y)
    {
```

```
        setWaypoint(x, y);

        if (check(x, y, 'N') && !(lastX == x && lastY == y - 1)) {
            if (x == t_x && y - 1 == t_y) {
                return true;
            }

            lastX = x;
            lastY = y;
            if (searchRec(x, y - 1))
                return true;
        }
        if (check(x, y, 'S') && !(lastX == x && lastY == y + 1)) {
            if (x == t_x && y + 1 == t_y) {
                return true;
            }

            lastX = x;
            lastY = y;
            if (searchRec(x, y + 1))
                return true;
        }
        if (check(x, y, 'E') && !(lastX == x + 1 && lastY == y)) {
            if (x + 1 == t_x && y == t_y) {
                return true;
            }

            lastX = x;
            lastY = y;
            if (searchRec(x + 1, y))
                return true;
        }
        if (check(x, y, 'W') && !(lastX == x - 1 && lastY == y)) {
            if (x - 1 == t_x && y == t_y) {
                return true;
            }

            lastX = x;
            lastY = y;
            if (searchRec(x - 1, y))
                return true;
        }

        unsetWaypoint(x, y);
        return false;
    }
```