```java
package se.kth.castor;

import org.apache.commons.math3.fraction.BigFraction;

public class NoteG {
    BigFraction[] v;
    int n;
    Printer p;

    public NoteG(int numberVar, int n, Printer p) {
        this.p = p;
        v = new BigFraction[numberVar];
        this.n = n;
        for (int i = 0; i < numberVar; i++) {
            v[i] = new BigFraction(0);
        }
        v[1] = new BigFraction(1);
        v[2] = new BigFraction(2);
        v[3] = new BigFraction(1);
    }

    public Number run() {
        double r = 0;
        int i = 0;
        // outer loop (compute B[2n - 1])
        while (true) {

            // pseudo-block to permit "break"
            while (true) {

                // 0: set index register
                i = 1;

                // 1: v4 = v5 = v6 = 2n
                v[4] = v[2].multiply(v[3]);
                v[5] = v[2].multiply(v[3]);
                v[6] = v[2].multiply(v[3]);
                p.print(v);
                // 2: v4 = 2n - 1
                v[4] = v[4].subtract(v[1]);
                p.print(v);
                // 3: v5 = 2n + 1
                v[5] = v[5].add(v[1]);
                p.print(v);
                // 4: v11 = (2n - 1)/(2n + 1) (the diagram seems to say v[5] / v[4]) [FIX]
                v[11] = v[4].divide(v[5]);
                p.print(v);
                // 5: v11 = (1/2) (2n - 1)/(2n + 1)
                v[11] = v[11].divide(v[2]);
```

```
p.print(v);
// 6: v13 = -(1/2) (2n - 1)/(2n + 1) = A0
v[13] = v[13].subtract(v[11]);
p.print(v);
// 7: v10 = n - 1
v[10] = v[3].subtract(v[1]);
p.print(v);

// branch if zero to operation 24
if (v[10].longValue() != 0) {

    // 8: v7 = 2
    v[7] = v[2].add(v[7]);
    p.print(v);
    // 9: v11 = (2n)/2 = A1 [why not just set v[11] = v[3] instead of 8 & 9]
    v[11] = v[6].divide(v[7]);
    p.print(v);
    // 10: v12 = A1 * B1
    v[12] = v[20 + i].multiply(v[11]);
    p.print(v);
    i = i + 1;
    // 11: v13 = A0 + A1 * B1
    v[13] = v[12].add(v[13]);
    p.print(v);
    // 12: v10 = n - 2
    v[10] = v[10].subtract(v[1]);
    p.print(v);

    // for each computed result, B = B3 [1], B5 [2], ...
    while (v[10].longValue() != 0) {
        // 13: v6 = 2n - 1 [1], 2n - 3 [2], ...
        v[6] = v[6].subtract(v[1]);
        p.print(v);
        // 14: v7 = 3 [1], 5 [2], ...
        v[7] = v[1].add(v[7]);
        p.print(v);
        // 15: v8 = (2n - 1)/3 [1], (2n - 3)/5 [2], ...
        v[8] = v[6].divide(v[7]);
        p.print(v);
        // 16: v11 = (2n)/2 * (2n - 1)/3 [1], (2n)/2 * (2n - 1)/3 * (2n - 3)/5 [2], ...
        v[11] = v[8].multiply(v[11]);
        p.print(v);
        // 17: v6 = 2n - 2 [1], 2n - 4 [2], ...
        v[6] = v[6].subtract(v[1]);
        p.print(v);
        // 18: v7 = 4 [1], 6 [2], ...
        v[7] = v[1].add(v[7]);
        p.print(v);
        // 19: v9 = (2n - 2)/4 [1], (2n - 4)/6
```

```java
                v[9] = v[6].divide(v[7]);
                p.print(v);
                // 20: v11 = (2n)/2 ∗ (2n − 1)/3 ∗ (2n − 2)/4 = A3 [1], (2n)/2 ∗ (2n − 1)/3 ∗
                // (2n − 2)/4 ∗ (2n − 3)/5 ∗ (2n − 4)/6 = A5 [2], ...
                v[11] = v[9].multiply(v[11]);
                p.print(v);
                // 21: v12 = A3 ∗ B3 [1], A5 ∗ B5 [2], ...
                v[12] = v[20 + i].multiply(v[11]);
                p.print(v);
                i = i + 1;

                // 22: v13 = A0 + A1 ∗ B1 + A3 ∗ B3 [1], A0 + A1 ∗ B1 + A3 ∗ B3 + A5 ∗ B5 [2],
                // ...
                v[13] = v[12].add(v[13]);
                p.print(v);
                // 23: v10 = n − 3 [1], n − 4 [2], ...
                v[10] = v[10].subtract(v[1]);
                p.print(v);
            } // while (v[10].longValue() != 0);
                // branch if non−zero to operation 13

            // terminate the pseudo−block
        }
        // 24: result (−v13) is copied into the results
        v[20 + i] = v[20 + i].subtract(v[13]);
        p.print(v);

        if (i == n)
            return v[20 + i];

        // 25: increase n, and reset working variables
        v[3] = v[1].add(v[3]);
        v[7] = new BigFraction(0);
        v[13] = new BigFraction(0);
        p.print(v);
    }
  }
 }
}
```