```
int m, n = 5

main()
{
    get m, n
    if m > n then
        print g()
    else
        print f()
}

f()
{
    int m
    m = n + m

    if g() = m then
        return f()^2
    else
        return g()
}

g()
{
    n++

    if f() = m then
        return g()*m
    else
        return f()-n
}
```

Nr.	Code	Comment
0	SET 0, 4	
1	SET 1, 6	
2	SET 3, 5	initialize n
3	SET 2, READ	get m
4	SET 3, READ	get n
5	JUMPT 14, D[2] ≤ D[3]	
6	SET 1, D[1] + 1	Calling sequence starts
7	SET D[1], 12	
8	SET D[1] + 1, D[0]	
9	SET 0, D[1]	
10	SET 1, D[1] + 2	
11	JUMP 43	Code segment for g(.), calling sequence ends
12	SET WRITE, D[D[1]]	print g(.)
13	JUMP 21	
14	SET 1, D[1] + 1	Calling sequence starts
15	SET D[1], 20	
16	SET D[1] + 1, D[0]	
17	SET 0, D[1]	
18	SET 1, D[1] + 4	
19	JUMP 22	Code segment for f(.), calling sequence ends
20	SET WRITE, D[D[1]]	print f(.)
21	HALT	
22	SET D[0]+2, D[3]+D[D[0]+2]	Code segment for f(.) starts
23	SET 1, D[1] + 1	Calling sequence starts
24	SET D[1], 29	
25	SET D[1] + 1, D[0]	
26	SET 0, D[1]	
27	SET 1, D[1] + 2	
28	JUMP 43	Code segment for g(.), calling sequence ends
29	SET D[0]+3, D[D[1]]	
30	JUMPT 41, D[D[0]+3]<>D[D[0]+2]	
31	SET 1, D[1] + 1	Calling sequence starts
32	SET D[1], 37	
33	SET D[1] + 1, D[0]	
34	SET 0, D[1]	
35	SET 1, D[1] + 4	
36	JUMP 22	Code segment for f(.), calling sequence ends
37	SET D[0]-1, D[D[1]]	
38	SET 0, D[D[0]+1]	Return sequence
39	SET 1, D[0]-1	
40	JUMP D[D[1]+1]	
41	SET D[0]-1, D[D[0]+3]	
42	JUMP 38	
43	etc…	Code segment for g(.) starts