Contents

1 Basic Test Results

```
********** FOLDER STRUCTURE TEST START **********
Extracting submission...
     Extracted zip successfully

Finding usernames...
     Submission logins are: nogafri
     Is this OK?

Checking for non-ASCII characters with the command 'grep -IHPnsr [^\x00-\x7F] <dir>' ...
     No invalid characters found.

**********  FOLDER STRUCTURE TEST END  **********


********** PROJECT TEST START **********
Running Mult test:
     Mult prog passed.

Running Fill test:
     Fill prog passed.

**********  PROJECT TEST END  **********


**************************************************
**********   PRESUBMISSION TESTS PASSED   **********
**************************************************

Note: the tests you see above are all the presubmission tests
for this project. The tests might not check all the different
parts of the project or all corner cases, so write your own
tests and use them!
```

2 AUTHORS

```
1 nogafri
2 Partner 1: Noga Friedman, noga.fri@mail.huji.ac.il, 209010479
3 Remarks:
```

3 fill/Fill.asm

```
// This file is part of nand2tetris, as taught in The Hebrew University, and
// was written by Aviv Yaish. It is an extension to the specifications given
// [here](https://www.nand2tetris.org) (Shimon Schocken and Noam Nisan, 2017),
// as allowed by the Creative Common Attribution-NonCommercial-ShareAlike 3.0
// Unported [License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

// This program illustrates low-level handling of the screen and keyboard
// devices, as follows.
//
// The program runs an infinite loop that listens to the keyboard input.
// When a key is pressed (any key), the program blackens the screen,
// i.e. writes "black" in every pixel;
// the screen should remain fully black as long as the key is pressed.
// When no key is pressed, the program clears the screen, i.e. writes
// "white" in every pixel;
// the screen should remain fully clear as long as no key is pressed.
//
// Assumptions:
// Your program may blacken and clear the screen's pixels in any spatial/visual
// Order, as long as pressing a key continuously for long enough results in a
// fully blackened screen, and not pressing any key for long enough results in a
// fully cleared screen.
//
// Test Scripts:
// For completeness of testing, test the Fill program both interactively and
// automatically.
//
// The supplied FillAutomatic.tst script, along with the supplied compare file
// FillAutomatic.cmp, are designed to test the Fill program automatically, as
// described by the test script documentation.
//
// The supplied Fill.tst script, which comes with no compare file, is designed
// to do two things:
// - Load the Fill.hack program
// - Remind you to select 'no animation', and then test the program
//   interactively by pressing and releasing some keyboard keys

// Put your code here.

// the program holds three variables:
// R0 - current screen address to color
// R1 - address of right after the end of the screen
// R2 - color to fill in {white = 0, black = -1}

(INIT)
    // init R0
    @SCREEN
    D=A
    @R0
    M=D
    // init R1
    @24576
    D=A
    @R1
    M=D
    // init R2 to white as default
    @R2
    M=0

```

```
(CHECK)  // check if user is pressing any key
    @KBD
    D=M
    @WHITE
    D;JEQ  // if D=0 fill in white
    @BLACK
    0;JMP  // else fill in black

(FILL)
    @R2  // get the color to fill with
    D=M
    @R0  // access current screen pixel address
    A=M
    M=D  // fill

    @R0
    M=M+1  // increment current pixel address by 1
    D=M  // holds R0
    @R1
    D=D-M  // R0-R1 (R0-24576)
    @INIT
    D;JEQ  // if reached the end of the screen (address 8192) restart the program
    @FILL
    0;JMP // else continue filling

(BLACK)
    @R2
    M=-1
    @FILL
    0;JMP

(WHITE)
    @R2
    M=0
    @FILL
    0;JMP
```

4 mult/Mult.asm

```
1  // This file is part of nand2tetris, as taught in The Hebrew University, and
2  // was written by Aviv Yaish. It is an extension to the specifications given
3  // [here](https://www.nand2tetris.org) (Shimon Schocken and Noam Nisan, 2017),
4  // as allowed by the Creative Common Attribution-NonCommercial-ShareAlike 3.0
5  // Unported [License](https://creativecommons.org/licenses/by-nc-sa/3.0/).
6
7  // Multiplies R0 and R1 and stores the result in R2.
8  //
9  // Assumptions:
10 // - R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.
11 // - You can assume that you will only receive arguments that satisfy:
12 //   R0 >= 0, R1 >= 0, and R0*R1 < 32768.
13 // - Your program does not need to test these conditions.
14 //
15 // Requirements:
16 // - Your program should not change the values stored in R0 and R1.
17 // - You can implement any multiplication algorithm you want.
18
19 // Put your code here.
20
21 // initialize values
22     @i
23     M=0  // i = 0
24     @R2
25     M=0  // R2 = 0
26
27 // if one of the factors is 0, stop and end with (R2 = 0)
28     @R0
29     D=M
30     @END
31     D;JEQ
32
33     @R1
34     D=M
35     @END
36     D;JEQ
37
38 // add R1 to R2 until i > R0
39 (LOOP)
40     @i
41     M=M+1
42     // if (i>R0) goto END
43     D=M
44     @R0
45     D=D-M // D = i - R0
46     @END
47     D;JGT
48
49     @R1
50     D=M
51     @R2
52     M=M+D // D = R1 + R2
53
54     @LOOP
55     0;JMP
56
57 // infinite loop
58 (END)
59     @END
```

```
60          0;JMP
```

5 swap/Swap.asm

```
// This file is part of nand2tetris, as taught in The Hebrew University, and
// was written by Aviv Yaish. It is an extension to the specifications given
// [here](https://www.nand2tetris.org) (Shimon Schocken and Noam Nisan, 2017),
// as allowed by the Creative Common Attribution-NonCommercial-ShareAlike 3.0
// Unported [License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

// The program should swap between the max. and min. elements of an array.
// Assumptions:
// - The array's start address is stored in R14, and R15 contains its length
// - Each array value x is between -16384 < x < 16384
// - The address in R14 is at least >= 2048
// - R14 + R15 <= 16383
//
// Requirements:
// - Changing R14, R15 is not allowed.

// Put your code here.

@R14
D=M
@minid  // init min variable as first element's address
M=D
@maxid  // init max variable as first element's address
M=D
@i  // init index variable
M=1
@curid  // init current element address as 0
M=0

(TRAVERSE)
    @i
    D=M
    @R15  // array length
    D=D-M
    @SWAP  // if reached end of array goto SWAP
    D;JEQ

    @R14
    D=M
    @i
    A=D+M
    D=A
    @curid
    M=D  // current location in array

    // check min:
    @curid
    A=M
    D=M
    @minid
    A=M
    D=M-D
    @SWITCHMIN  // switch if current element is smaller
    D;JGT

    // check max:
    @curid
    A=M
    D=M
```

```
60      @maxid
61      A=M
62      D=D-M
63      @SWITCHMAX  // switch if current element is bigger
64      D;JGT
65
66      @i
67      M=M+1
68      @TRAVERSE
69      0;JMP
70
71  (SWITCHMIN)
72      @curid
73      D=M /// QUESTION - DOES IT MATTER IF I WRITE HERE " D=A "?
74      @minid
75      M=D
76      @i
77      M=M+1
78      @TRAVERSE
79      0;JMP
80
81  (SWITCHMAX)
82      @curid
83      D=M
84      @maxid
85      M=D
86      @i
87      M=M+1
88      @TRAVERSE
89      0;JMP
90
91  (SWAP)
92      // switch min and max values:
93      @maxid
94      A=M
95      D=M
96      @maxval
97      M=D
98
99      @minid
100     A=M
101     D=M
102     @minval
103     M=D
104
105     @minval
106     D=M
107     @maxid
108     A=M
109     M=D
110
111     @maxval
112     D=M
113     @minid
114     A=M
115     M=D
116
117     @END
118     0;JMP
119
120 (END)
121     @END
122     0;JMP
```