Contents

1 Basic Test Results

```
********** TESTING FOLDER STRUCTURE START **********
Checking your submission for presence of invalid (non-ASCII) characters...
No invalid characters found.
Submission logins are: linorcohen
Is this OK?
**********  TESTING FOLDER STRUCTURE END  **********

********** PROJECT TEST START **********
Testing.
Fill.asm: passed the test
Mult.asm: passed the test
**********  PROJECT TEST END  **********

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

```
linorcohen
Partner 1: Linor Cohen, linor.cohen@mail.huji.ac.il, 318861226
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

(LOOP)
//  if (keyboard != 0)
    @KBD
    D=M
    @BLACK
    D;JNE

//  if (keyboard == 0)
    @KBD
    D=M
    @WHITE
    0;JMP

(BLACK)
    @cur
    M=-1
    @START
    0;JMP

(WHITE)
    @cur
    M=0
    @START
    0;JMP

(START)
//  i = 0
    @i
    M=0

//  dest = 8192
    @8192
    D=A
    @dest
    M=D

    (INNERLOOP)
//  if (i == dest) goto END
    @i
    D=M
    @dest
    D=D-M
    @LOOP
    D;JEQ

//  screen[i] = cur
    @i
    D=M
    @SCREEN
    D=D+A
    @address
    M=D
    @cur
    D=M
    @address
    A=M
    M=D

//  i = i + 1
    @i
    M=M+1
    @INNERLOOP
    0;JMP
(END)
@END
0;JMP
```

4 mult/Mult.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/).

// Multiplies R0 and R1 and stores the result in R2.
//
// Assumptions:
// - R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.
// - You can assume that you will only receive arguments that satisfy:
//   R0 >= 0, R1 >= 0, and R0*R1 < 32768.
// - Your program does not need to test these conditions.
//
// Requirements:
// - Your program should not change the values stored in R0 and R1.
// - You can implement any multiplication algorithm you want.

//i = 0
@i
M=0
@R2
M=0

(LOOP)
//  if (i == R1) goto END
    @i
    D=M
    @R1
    D=D-M
    @END
    D;JEQ

//  sum = sum + R0
    @R0
    D=M
    @R2
    M=D+M

//  i = i + 1
    @i
    M=M+1

//  goto LOOP
    @LOOP
    0;JMP

(END)
    @END
    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.

// i = 0
@i
M=0

// biggest = R14 + i (address)
    @i
    D=M
    @R14
    D=D+M
    @biggest
    M=D

// smallest = R14 + i (address)
    @i
    D=M
    @R14
    D=D+M
    @smallest
    M=D

(LOOP)
// if (i == R15) goto END
    @i
    D=M
    @R15
    D=D-M
    @END_LOOP
    D;JEQ

// if (array[i] > biggest)
    @i
    D=M
    @R14
    D=D+M
    A=D //load address for R14 + i
    D=M //load value for R14 + i
    @biggest
    A=M //load address for biggest
    D=D-M
    @SET_MAX
    D;JGT

// if (array[i] < smallest)
```

```
    @i
    D=M
    @R14
    D=D+M
    A=D //load address for R14 + i
    D=M //load value for R14 + i
    @smallest
    A=M //load address for biggest
    D=D-M
    @SET_MIN
    D;JLT

    @CONTINUE
    0;JMP

(SET_MIN)
//smallest = array[i].address
    @i
    D=M
    @R14
    D=D+M
    @smallest
    M=D
    @CONTINUE
    0;JMP


(SET_MAX)
//biggest = array[i].address
    @i
    D=M
    @R14
    D=D+M
    @biggest
    M=D

(CONTINUE)
//i = i + 1
    @i
    M=M+1
    @LOOP
    0;JMP

(END_LOOP)

// temp = min
@smallest
A=M
D=M
@R0
M=D
// min = max
@biggest
A=M
D=M
@smallest
A=M
M=D
// max = temp
@R0
D=M
@biggest
A=M
M=D

(END)
@END
0;JMP
```