Question Number 0

Write a Hack assembly program that **copies** the value from memory location 0 into memory location 2.

Solution.

- **Before Runnig program**

./images/q0_1.png

- **After Runnig program**

./images/q0_2.png

Question Number 1

Write a Hack assembly program that **subtracts** the values stored in memory locations 1 and 2, and stores the result in memory location 0.

Solution.

- **Before Runnig program**

./images/q1_1.png

- **After Runnig program**

./images/q1_2.png

Note
Here I am subtracting $Ram[0] = Ram[1] - Ram[2]$

Question Number 2

Write a Hack assembly program that **swaps** the values stored in memory locations 0 and 1.

Solution.

- **Before Runnig program**

./images/q2_1.png

- **After Runnig program**

./images/q2_2.png

Question Number 3

Write a Hack assembly program that **checks** if the value stored in memory location 0 equals that stored in memory location 1. If they are equal, store 1 in memory location 2; otherwise, store 0.

Solution.

- For **Not Equal** Numbers
  - **Before Runnig program**

    ./images/q3_1_1.png

  - **After Runnig program**

    ./images/q3_1_2.png

- For **Equal** Numbers
  - **Before Runnig program**

    ./images/q3_2_1.png

  - **After Runnig program**

    ./images/q3_2_2.png

Question Number 4

Write a Hack assembly program that implements a simple **loop** to increment the value in memory location 0 by 1 a total of 5 times, storing the result in memory location 1.

Solution.

- **Before Runnig program**

./images/q4_1.png

- **After Runnig program**

./images/q4_2.png

Question Number 5

Write a Hack assembly program that **reads from the keyboard** and stores the code of the first key at RAM[0] and code of the second key at RAM[1] and then adds the codes and stores at RAM[2]. after that it **blackens** the first 16 pixels of row 6 of the screen.

Solution.

- **Before Runnig program**

./images/q5_1.png

- **After 1 Key Press**

./images/q5_2.png

- **After 2 Key Press**

./images/q5_3.png

Question Number 6

Write a Hack assembly program that continuously **checks for keyboard input**. Whenever any key is pressed, the program should **black** the first 16 pixels of the top row(top left corner) of the screen. The program should keep running, waiting for additional key presses, and each key press should result in a black line being drawn on the screen.

Solution.

- **Before Runnig program**

./images/q6_1.png

- **After 10 Key Press**

./images/q6_10.png

- **After 25 Key Press**

./images/q6_25.png

Question Number 7
Mult

Solution.

- **Before Runnig program**

./images/q7_1.png

- **After running program using mult.tst file**

./images/q7_2.png

Question Number 8
Fill

Solution.

- **Before Runnig program**

./images/q8_1.png

- **After running program using FillAutomatic.tst file**

./images/q8_2.png

- **When no key pressed**

./images/q8_3.png

- **When key get pressed**

./images/q8_4.png