# Unit 7 Assignment Solution

In the first problem we needed to implement a 'for' loop using the assembler available in the Hack system. The example in Assembler has a lot of comments designed to help you understand the program. First, everything to the right of the // is actually a comment. The only code is actually to the left of the //. Immediately to the right of the //, there is a number. This represents the location in ROM of each line in the program. I have the heading "program counter" above this to indicate that if you watch the current state of the program counter in the CPU simulator you will see the code being executed for any program counter number matches the number in the comments.

One important thing to keep in mind is that these examples have been put together using a 'brute force' approach. What this means is that the most straight forward approach to solve the problem was used. It is possible (and likely) that there is a more elegant or efficient way to implement any of these problems. This was not done for these examples to make the code easier to read and follows. If you have implemented your solutions to these problems using a more elegant approach your solution is just as valid as these examples!

```
        // Problem 1 - for loop
        //
        //      J=5
        //      for(i=1; i<5; i++) {
        //          j--
        //      }
        //
        //Program
        //Counter
        //              initialize j variable with value of 5
@5      //00           Load value of 5 into A register
D=A     //01           Move A to D register
@0      //02           Load memory address 0
M=D     //03           Move D register to address 0
        //              initialize i variable with value of 1
@1      //04           Load value of 1 into A register
D=A     //05           Move A to D register
@1      //06           Load memory address 1
M=D     //07           Move D register to address 1
        //
        //Test i to make sure it is still less than 5
@1      //08           Load address 1
D=M     //09           Move value in address to D register
@5      //10           Load value of 5 into A register
D=D-A   //11           Subtract D register from A register and store in D
@25     //12           Load program counter 8
D;JGE   //13           Execute a jump if i is 5
        //
```

```
        //increment the value of i variable
@1      //15           Load memory address 1
D=M+1   //16           Decrement Memory address and store in D
@1      //17           Load memory address 1
M=D     //18           Store D in address 1
        //
        //decrement the value of j variable
@0      //19           Load memory address 0
D=M-1   //20           Decrement memory address and store in D
@0      //21           Load memory address 0
M=D     //22           Store D in address 0
        //
        //Execute unconditional jump to beginning of the loop
@8      //23           Load program counter address 8
D;JMP   //24           Execute a jump to 8
        //
        //Unless loop for end of program
@25     //25           Load program counter address 25
D;JMP   //26
```

**Problem 2** demonstrates the implementation of an if-then-else type of conditional structure.

```
        // Problem 2 - if then else
        //
        //      i=4
        //      if (i<5) then
        //         j=3
        //      else
        //         j=2
        //
        //Program
        //Counter
        //              initialize i variable with value of 4
@4      //00           Load value of 4 into A register
D=A     //01           Move A to D register
@0      //02           Load memory address 0
M=D     //03           Move D register to address 0
        //
        //Test i to determine if it is less than 5
@0      //04           Load address 1
D=M     //05           Move value in address to D register
@5      //06           Load value of 5 into A register
D=D-A   //07           Subtract D register from A register and store in D
@16     //08           Load program counter with 16
D;JLT   //09           Execute a jump if i < 5
        //
        //else initialize variable j to the value 2
```

```
@1      //10           Load value of 2 into A register
D=A     //11           Move A to D register
@1      //12           Load memory address 1
M=D     //13           Store D in address 1
@20     //14           Load program counter with x
D;JMP   //15           Execute an unconditional jump to end of routine
        //
        //initialize the variable j to the value of 3
@3      //16           Load value of 3 into A register
D=A     //17           Move A to D register
@0      //18           Load memory address 1
M=D     //19           Store D in address 1
        //
        //loop for end of program
@21     //20           Load program counter address 25
D;JMP   //21
```

**Problem 3** implements a while loop and within the while loop an if expression showing the use of complex nested code implemented using the Hack assembler.

```
        // Problem 3 - while loop
        //
        //      i = 0
        //      j = 0
        //      while(i==0) {
        //         j++
        //         if j == 5 then
        //          i = j
        //      }
        //
        //Program
        //Counter
        //              initialize i variable with value of 0
@0      //00           Load value of 0 into A register
D=A     //01           Move A to D register
@0      //02           Load memory address 0
M=D     //03           Move D register to address 0
        //
        //              initialize j variable with value of 0
@0      //04           Load value of 0 into A register
D=A     //05           Move A to D register
@1      //06           Load memory address 1
M=D     //07           Move D register to address 0
        //
        //Test i to determine if it is equal to 0
@0      //08           Load value of 0 into A register
D=A     //09           Move A to D register
```

```
@0      //10           Load memory address 0
D=D-M   //11           Subtract D register from A register and store in D
@31     //12           Load program counter 31
D;JNE   //13           Execute a jump if i is not equal to 0 (jump out of loop)
        //
        //increment the value of j variable
@1      //14           Load memory address 1
D=M+1   //15           Increment Memory value and store in D
@1      //16           Load memory address 1
M=D     //17           Store D in address 1
        //
        //Test j to determine if it is equal to 5
@5      //18           Load value of 0 into A register
D=A     //19           Move A to D register
@1      //20           Load memory address 1
D=D-M   //21           Subtract D register from A register and store in D
@8      //22           Load program counter 30
D;JNE   //23           Execute a jump if i is equal to 0 (jump out of loop)
        //
        //Assign the value of variable j to variable i
@1      //24           Load memory address 1
D=M     //25           Move memory to D register
@0      //26           Load memory address 0
M=D     //27           Move D register to Memory
@8      //28           Load Program counter 8
D;JMP   //29           Jumpt to 8 (beginning of the while loop)
        //
        //
@31     //30           Load program counter 31
D;JMP   //31           Execute jump which creates continuous loop
```

***Problem 4:*** demonstrates the ability to create an array and then manipulate the array both by traversing it (moving through the array to be able to extract items in the array) and also demonstrating the ability to update values or use values within the array.

One important feature that you will want to understand in this example solution appears in red in the following code. In this case we are maintaining the array index in memory location 10. What we want to do is use the current value of the index to get access to an item in the array. The way that we do this is by loading the value in memory location 10 into the A register. Any value in the A register when using an M command is assumed to be a memory address so these two lines of code in effect implement the ability to access items in the array via an index.

```
//Test A[i] to see if it is == 0
@10     //34           Load address 10
A=M     //35           Move the value in address 10 to A register
```

```
        // Problem 4 -load and traverse an array
        //
        //      A[0]=5
        //      A[1]=4
        //      A[2]=3
        //      A[3]=2
        //      A[4]=1
        //      A[5]=0
        //
        //      for (i=0; i<=5; i++) {
        //              if A[i] == 0 then
        //              A[i] = 5;
        //      }
        //
        //
        //Program
        //Counter
        //              initialize the array with 5 values
@5      //00           Load value of 5 into A register
D=A     //01           Move A to D register
@0      //02           Load memory address 0
M=D     //03           Move D register to address 0
@4      //04           Load value of 4 into A register
D=A     //05           Move A to D register
@1      //06           Load memory address 1
M=D     //07           Move D register to address 1
@3      //08           Load value of 3 into A register
D=A     //09           Move A to D register
@2      //10           Load memory address 2
M=D     //11           Move D register to address 2
@2      //12           Load value of 2 into A register
D=A     //13           Move A to D register
@3      //14           Load memory address 3
M=D     //15           Move D register to address 3
@1      //16           Load value of 1 into A register
D=A     //17           Move A to D register
@4      //18           Load memory address 4
M=D     //19           Move D register to address 4
@0      //20           Load value of 0 into A register
D=A     //21           Move A to D register
@5      //22           Load memory address 5
M=D     //23           Move D register to address 5
        //
        //              initialize i variable with value of 1
@0      //24           Load value of 0 into A register
```

```
D=A     //25           Move A to D register
@10     //26           Load memory address 10
M=D     //27           Move D register to address 10
        //
        //Test i to make sure it is still less than 6
@10     //28           Load address 10
D=M     //29           Move value in address to D register
@6      //30           Load value of 6 into A register
D=D-A   //31           Subtract D register from A register and store in D
@48     //32           Load program counter 48 - end of loop
D;JGE   //33           Execute a jump if i is 6
        //
        //Test A[i] to see if it is == 0
@10     //34           Load address 10
A=M     //35           Move the value in address 10 to A register
D=M     //36           Move Memory to D register where we test for 0
@44     //37           Load program counter with
D;JNE   //38           Jump if not equal to 0
        //
        //If the value of A[i] == 0 then assign 5 to it
@5      //39           Load value of 5 to A register
D=A     //40           Move to D register
@10     //41           Load address 10 (location of variable i)
A=M     //42           Move the value in address 10 to A register
M=D     //43           Move D register to memory
        //
        //increment the value of i variable
@10     //44           Load memory address 10
M=M+1   //45           Increment Memory address and store in D
@28     //46           Load address 28 (beginning of loop)
D;JMP   //47           Jump to beginning of the loop
@48     //48           Load address 53
D;JMP   //49           Execute unconditional Jump
```