ICS Lab Report #5

StuID: **Name:**

Problem Setting

Here we are using the LC3 assembly language to solve a classic problem in dynamic programming. One can start at any point on a 2 dimensional map with height data, and can "ski" to a neighbor position with a lower height. We should calculate the longest route and store the length in R2.

Algorithm Specification

As usually an alternative to DP methods, we exploit memorized search.

The main program clear a VISIT array, and try to search for the result by each position.

```
clear VISIT[M][N]
ans <- 1
for i = 1 : m
  for j = 1 : n
    ans <- max(ans, memsear(i, j) - 1)
```

memsear() is used to get the longest path by exploring 4 directions.

```
if position out of map
  return 1
ARRAY[x][y] <- 0
a <- ARRAY[x][y]
if VISIT[x][y]
  return a+1
else
  VISIT[x][y] <- 1
a <- max(a, memsear(x, y+1))
a <-max(a, memsear(x, y-1))
a <-max(a, memsear(x+1, y))
a <-max(a, memsear(x-1, y))
VISIT[x][y] <- a
return a+1
```

LC3 Implementation

Main program can be easily implemented so ignored. Here we give out the memsear() function:

```
MSEAR
...... ;store registers
......
BRZP RETURN_FAIL ;check out of bound. If failed, return 1.

AND R2, R2, #0
LDI R3, ADDR_N
AND R4, R4, #0
ADD R4, R4, R0
BRZ OUTLOOPMUL
LOOPMUL
ADD R2, R2, R3
ADD R4, R4, #-1
BRP LOOPMUL
OUTLOOPMUL
ADD R2, R2, R1
LD R3, ADDR_ARRAY
ADD R3, R2, R3
LDR R3, R3, #0
LDR R4, R6, #6
NOT R4, R4
ADD R4, R4, #1
ADD R4, R3, R4
BRZP RETURN_FAIL ;check height lower than previsited height
LD R3, VISIT
ADD R3, R2, R3
LDR R4, R3, #0
ADD R4, R4, #0
BRP RETURN_VISITED ;if visited, return the answer
ADD R4, R4, #1
STR R4, R3, #0

ADD R6, R6, #-1
LD R3, ADDR_ARRAY
ADD R3, R2, R3
```

```
LDR R3, R3, #0
STR R3, R6, #0
LD R3, DP
ADD R3, R2, R3
AND R2, R2, #0
ADD R2, R2, #1
STR R2, R3, #0
ADD R0, R0, #1
JSR MSEAR
LDR R2, R3, #0
NOT R2, R2
ADD R2, R2, #1
ADD R4, R2, R5
BRNZ CHECK2
STR R5, R3, #0
CHECK2
ADD R0, R0, #-2
JSR MSEAR
LDR R2, R3, #0
NOT R2, R2
ADD R2, R2, #1
ADD R4, R2, R5
BRNZ CHECK3
STR R5, R3, #0
CHECK3
ADD R0, R0, #1 ;search in 2 directions, update ans
...... ;similar as above
...... ;restore stack
RET
```

Check Problem

Q: How many elements will be in your stack at most?

A: 50(map size, thus max recursion size) * 7(each iteration push 6 registers and 1 height datum) = 350 elements.