2021 ICS-lab5

Main Algorithm

First allocate a memory 60-word space, and initialize it to be all -1.This space is used to be the memory space to save the longest length of each point. Then load the width and the height, then do a travsal on the map. If a point has been visited, then we just load it from the memory space above. If not, we do a depth-first search on its four directions to get the longest path and save it to the stack when the subroutine return. And each time we call the dfs function we compare the return value with the result we save, if it's bigger, we change the result to be the value. Finally we save the result to the R2, halt the machine.

Essential Parts

```
;  main part
...
initializemem ;used to initialized to memory space we used the save the result to each
point.
    brz initdone
    str r3,r2, #0
    add r2,r2, #1
    add r1,r1, #-1
    brnzp initializemem
...
innerloop
...
add r6,r6, #-2
str r1,r6, #0
str r2,r6, #1; load the data for subroutine
st r1,nowrow
st r2,nowcolumn ;save current data
JSR dfs ; the function we call will store the biggest length for it on the top of the
stack
ld r1, nowrow ; restore current data
ld r2, nowcolumn
ldr r3,r6, #0
add r6,r6, #1
add r5,r6, #0
ld r7, result
not r7,r7
add r7,r7, #1
add r7,r3,r7 ; get and compare the data the subroutine returned and compare it with our
presaved result.
brp change
...
nextloop
...
```

```
ld r2,result
add r2,r2, #1 ; the loop ends, we save the result to r2
halt



;  dfs part:
    ... ;; save r7 and r5
    ldr r0,r5,#1
    ld r1, fullcolumn
    and r2,r2, #0
    add r0,r0,#0
mul brz mulover
    ... ; used to get the relative position to the head of the map
mulover
    ldr r1,r5, #2
    add r2,r2, r1
    add r2,r2, #2 ; r2 -> pos
    lea r7,memarea
    add r7,r2,r7
    ldr r3,r7, #0 ;; to see if this point has been memorized
    brzp ismem ; if it is, then we go to the end of this function.
    ...
left
    ldr r0,r5,#1
    ldr r1,r5,#2 ; to see if it cannot go to the right
    brz notleft ; push 0
    ldr r2, r5, #-2
    add r2, r2, #-1
    ld r3,maphead
    add r3,r3,r2
    ldr r3,r3, #0
    not r3,r3
    add r3,r3, #1
    ldr r4,r5, #-3
    add r3,r3,r4 ; to see if it's bigger then right
    brnz notleft
    ldr r0,r5, #1
    ldr r1,r5, #2
    add r6, r6 ,#-2
    str r0,r6, #0
    add r1,r1, #-1
    str r1, r6, #1
    JSR dfs
    ldr r7,r6, #0
    add r7,r7, #1
    str r7,r6, #0 ;get the result and push it to stack
    brnzp right
notleft ;; if it cannot go to right, we just push 0 to the stack
```

```
    add r6,r6, #-1
    and r0,r0, #0
    str r0,r6, #0
right
   ... ; just similiar to the left
notright
   ... ; the same
up
  ...
notup
  ...
down
  ...
notdown
  ...
;; compare four values we push above, and leave the biggest to the stack
 ... ;compare part
finish
  ; pop temp value, save the value we get above to the memory area, and return
ismem
  ; it has been memoried, so we just load the data and push the data to the stack ,
restore r7 and r5, return.
```

Questions:

No question.