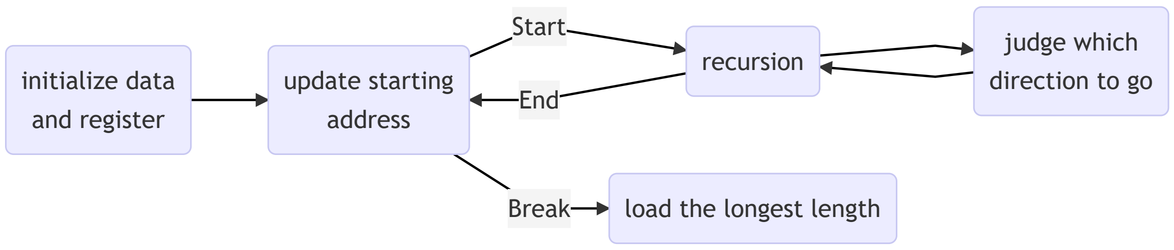
Lab5 Report:

Mermaid:



Code:

```
;this is part of program code and west,south,east is the same as north
CHECK_NORTH JSR CHECK_NORTH_FRAME       ;check whether there is frame in the north
            BRz CHECK_EAST              ;if there is frame in the north ,then go to CHECK_EAST
            JSR CHECK_GO                ;if there isn't frame in the north ,then we check data of R0 and data of R3
            ADD R1,R1,#0
            BRnz CHECK_EAST             ;if R0_data > R3_data ,then we go to CHECK_EAST
            ADD R2,R2,#1                ;current length++;
            JSR CHECK_BIG               ;update the biggest length
            JSR BREAD                   ;leave breadcrump in current address
            JSR DFS
            JSR FLIP_BREADCRUMP         ;clear breadcrump in current address
            ADD R2,R2,#-1               ;current length--;
;this function is to check the boundry of maze
CHECK_NORTH_FRAME   LD  R1,FLIP_LENGTH          ;
                    ADD R0,R0,R1
                    LDR R1,R0,#0
                    RET
;this function is to check whether we could go in this direction
CHECK_GO        ADD R6,R6,#-1
                STR R2,R6,#0

                LDR R1,R0,#0
                LDR R2,R3,#0
                NOT R1,R1
                ADD R1,R1,#1
                ADD R1,R1,R2

                LDR R2,R6,#0
                ADD R6,R6,#1
                RET
```

```
; this function is to place a flag in current address
BREAD           LD  R1,BREADCRUMB
                ADD R4,R4,R1
                STR R4,R3,#0
                RET
;this function is to when we is backtracking, we should clear the flag
FLIP_BREADCRUMP LD  R1,FLIP_BREAD
                ADD R4,R4,R1
                STR R4,R3,#0
                RET
;function check_big is to store the longest length into register
;we should initialize all the data before recursion DFS
```

Algorithm:

1. Firstly, we initialize all the data and load maze into the program.
2. Secondly, we begin recursion. And we check north first, if we could go to north, then we leave breadcrump in this address and start next recursion. If four direction are all blocked, then we begin to backtrack and clear the breadcrump in this address.
3. After traverse all the address, then we load the longest length to R2.
4. (The main principle is the same with DFS and pay attention to pop, push, and details of backtracking
5. There are two factors whether we could go to north successfully. One is number in the north should lower than current address and the other is there is no frame in the north.

Check:

- TA: How do you prevent professor patt skiing in a circle?

  Me: Because if A>B>C>D, there is no chance that A<D. Thus, there is no chance.