Lab5-Report

1 Algorithm

Main Function

- Calculate the map size
- Iterate from 0 to mapsize as the start position
- Call the recursive function
- Memorize the longest distance from the current cell

Recursive Function

- Use stack to save modified registers
- Update the current cell information
- Check whether or not the current cell was visited.If so, jump to the comparison part.
- Check the north cell. First check if we can go north, then check if the height of the northern cell is lower than that of the current cell. Otherwise, go south.
- Similarly, we can check the south, east and west cell.
- Finally, compare the pedometer with the longest distance. If the pedometer is larger, update the longest distance.
- Use stack pop to restore registers.

2 Codes & Comments

```
; Recursive Subroutine to find the longest distance when skiing
; input: R0, the current cell address
; output: Global, the current longest distance

                ; save modified registers into the stack
Skiing          ADD         R6, R6, #-1
                STR         R1, R6, #0              ; R1 holds the cell height of
caller
                ADD         R6, R6, #-1
                STR         R2, R6, #0              ; R2 holds the Pedometer
                ADD         R6, R6, #-1
                STR         R3, R6, #0              ; R3 holds the pointer of memorized
map
                ADD         R6, R6, #-1
                STR         R4, R6, #0              ; R4 holds the cell address of
caller
                ADD         R6, R6, #-1
                STR         R5, R6, #0              ; R5 holds the main program
iterative counter
                ADD         R6, R6, #-1
                STR         R7, R6, #0              ; R7 holds the PC of caller
```

```
                ; update current cell information

                LDR         R1, R0, #0              ; R1 now holds the current cell
height
                ;
VISITED         LD          R7, NegBase
                ADD         R7, R7, R0
                LEA         R3, Memorize
                ADD         R3, R3, R7
                LDR         R7, R3, #0
                BRnz        CHECK_NORTH
                ADD         R2, R2, R7
                ADD         R2, R2, #-1
                BR          CmpDistance
                ;
CHECK_NORTH     ADD         R4, R0, #0              ; R4 now holds the current cell
address
                LD          R3, NegColumn
                ADD         R3, R0, R3
                LD          R7, NegBase
                ADD         R7, R7, R3
                BRn         CHECK_SOUTH             ; edge check, cannot go north: R0-
MapBase-Column<0
                LDR         R5, R3, #0
                NOT         R5, R5
                ADD         R5, R5, #1
                ADD         R5, R5, R1
                BRnz        CHECK_SOUTH             ; height check, cannot go north: R1
- mem[R0-column] <= 0
                ADD         R0, R3, #0
                ADD         R2, R2, #1
                JSR         Skiing
                ADD         R0, R4, #0
                ADD         R2, R2, #-1             ; restore the current address and
pedometer
                ;
; Check south, west and east, which are omitted here.
                ; arrive at the lowest cell, compare the current pedometer with the
longest distance
                ; if the current pedometer is larger, than update the longest
CmpDistance     LD          R3, CurLength
                NOT         R3, R3
                ADD         R3, R3, #1
                ADD         R3, R3, R2
                BRnz        CmpLongest
                ST          R2, CurLength
CmpLongest      LD          R3, Longest
                NOT         R3, R3
                ADD         R3, R3, #1
```

```
                ADD         R3, R3, R2
                BRnz        RESTORE
                ST          R2, Longest
                ;
RESTORE         ADD         R0, R4, #0
                LDR         R7, R6, #0
                ADD         R6, R6, #1
                LDR         R5, R6, #0
                ADD         R6, R6, #1
                LDR         R4, R6, #0
                ADD         R6, R6, #1
                LDR         R3, R6, #0
                ADD         R6, R6, #1
                LDR         R2, R6, #0
                ADD         R6, R6, #1
                LDR         R1, R6, #0
                ADD         R6, R6, #1
                RET
                .END
```

3 TA's Questions

3.1 Explain your recursive subroutine

- Use stack to save modified registers
- Update the current cell information
- Check whether or not the current cell was visited.If so, jump to the comparison part.
- Check the north cell. First check if we can go north, then check if the height of the northern cell is lower than that of the current cell. Otherwise, go south.
- Similarly, we can check the south, east and west cell.
- Finally, compare the pedometer with the longest distance. If the pedometer is larger, update the longest distance.
- Use stack pop to restore registers.

3.2 How to use stack to save and restore?

- In the main program, we must set the user stack, otherwise R6 is random, which may cause ACV.
- At the beginning of the recursive subroutine, we must use PUSH to save R1, R2, R3, R4, R5, R7.
- At the end of the subroutine, we must use POP to save R7, R5, R4, R3, R2, R1 reversely.