NAME: Akintayo Lanre Moshood

MATRIC NO: 222460

GITHUB REPO: https://github.com/LanreAkintayo/adam-project-on-c

Open the file labelled Adam programming language.

i. Use a pseudocode to analyse the problem.
ii. Write a C programming language to solve the problem.
The code can be found in the github repo above.

Pseudocode

```
function getCaseLength(){
    Declare noOfCases

    fp <- open(“adam.in”) in read only

    if fp is null {
        DISPLAY(“unable to open file”)
        EXIT
    }

    line ← null
    len ← 0

    if ((READ FIRST LINE INTO VARIABLE line AND len) is -1){
          noOfCases ← line
    }

    CLOSE(fp)
    FREE(line)

    RETURN noOfCases
}


function checkCaseValidity(scenario) {
    DECLARE result
    DECLARE expression

    result ← REGULAR_EXPRESSION(expression, “[^udUD]”, 0)
    result ← REG_EXPRESSION_EXECUTE(expression, case, 0, NULL, 0)

    RETURN result
}

function readCases(noOfCases, cases){
    fp <- open(“adam.in”) in read only

    if fp is null {
        DISPLAY(“unable to open file”)
        EXIT
    }

    line ← null
    len ← 0
    counter ← 0

    while ((READ FIRST LINE INTO VARIABLE line AND len) is -1){
        if counter is 0 {
            if ((READ FIRST LINE INTO VARIABLE line AND len) is -1){
                OVERRIDE \n to 0 in line
                COPY line at index counter to cases
            }
        } ELSE {
            OVERRIDE \n to 0 in line
            COPY line at index counter to cases
        }

        if counter is noOfCases {
            BREAK
        }

        INCREMENT counter

        END WHILE

    }

    CLOSE(fp)
    FREE(line)
}

function getNoOfSteps(_case){
    counter ← 0

    result <- CALL checkCaseValidity(_case)

    if result is 0 {
        counter ← -1
        RETURN counter
    }

    for (i = 1 to LENGTH_OF(_case)){
        currentChar ← ELEMENT AT INDEX i FROM _case

        if TOUPPERCASE(currentChar) is 68{
            BREAK
        }

        INCREMENT counter BY 1
    }

    RETURN counter
}

function displayResult(noOfCases, cases){
    for (i = 0 to noOfCases){
        noOfSteps <- CALL getNoOfSteps(ELEMENT AT INDEX i FROM cases)
        result <- CALL checkCaseValidity(ELEMENT AT INDEX i FROM cases)

        if (result is 0){
```

```
                 DISPLAY(“\nInvalid steps found in: <ELEMENT AT INDEX i FROM cases> Step can
                 either be 'U' or 'D'”)
          } ELSE{
                 DISPLAY(“No of steps for <ELEMENT AT INDEX i FROM cases> is <noOfSteps> ”)

          }
    }

}


function main(){
    noOfCases <- CALL getCaseLength()
    DECLARE cases

    CALL readCases(noOfCases, cases)
    CALL displayResult(noOfCases, cases)

    RETURN 0
}
```