Week8 BFL

```
DECLARE colors: ARRAY[1:100000] OF STRING
DECLARE n: INTEGER

INPUT n

DECLARE i:INTEGER
FOR i <- 1 TO n
    INPUT colors[i]
NEXT i

FOR i <- n TO 1 STEP -1
    IF colors[i] = "green" THEN
        DECLARE j:INTEGER
        FOR j <- i TO n
            colors[j] <- colors[j+1]
        NEXT j
        n <- n-1
    ENDIF
NEXT i

FOR i <- 1 TO n
    OUTPUT colors[i]
NEXT i
```

```
DECLARE MyNumbers : ARRAY[1:10] OF INTEGER
DECLARE Squares : ARRAY[1:10] OF INTEGER

DECLARE i : INTEGER
FOR i <- 1 TO 10
    MyNumbers[i] <- i
NEXT i
FOR i <- 1 TO 10
    Squares[i] <- MyNumbers[i] * MyNumbers[i]
NEXT i
FOR i <- 1 TO 10
```

```
    OUTPUT Squares[i]
NEXT i
```

```
CONSTANT N = 10
DECLARE MyNumbers : ARRAY[1:N] OF INTEGER
DECLARE Odd : INTEGER

DECLARE i : INTEGER
FOR i <- 1 TO N
    INPUT MyNumbers[i]
NEXT i

FOR i <- 1 TO N
    IF MyNumbers[i] MOD 2 = 1 THEN
        Odd <- Odd + 1
    ENDIF
NEXT i
OUTPUT "Odd:", Odd, "Even", N-Odd
```

```
CONSTANT N = 25.0
DECLARE Ages : ARRAY[1:N] OF REAL
Ages <- [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
DECLARE Highest:INTEGER
DECLARE Lowest:INTEGER
DECLARE Average:REAL

Highest <- 0
Lowest <- 200
Average <- 0
DECLARE i:INTEGER
FOR i <- 1 TO N
    IF Ages[i] > Highest THEN
        Highest <- Ages[i]
    ENDIF
    IF Ages[i] < Lowest THEN
        Lowest <- Ages[i]
    ENDIF
    Average <- Average + Ages[i]/N
NEXT i
```

```
OUTPUT "Highest", Highest, "Lowest", Lowest, "Average", Average
```

```
// Global Arrays
DECLARE Elements: ARRAY[0:4, 0:2] OF STRING

Elements[0] <- ["Antimony", "Sb", "Stibium"]
Elements[1] <- ["Copper", "Cu", "Cuprum"]
Elements[2] <- ["Gold", "Au", "Aurum"]
Elements[3] <- ["Iron", "Fe", "Ferrum"]
Elements[4] <- ["Lead", "Pd", "Plumbum"]

DECLARE i:INTEGER
DECLARE j:INTEGER
DECLARE tmp:STRING
FOR i <- 0 TO 4
    tmp <- ""
    FOR j <- 0 TO 2
        tmp <- tmp & " " & Elements[i][j]
    NEXT j
    OUTPUT tmp
NEXT i
```

```
CONSTANT ClassSize = 20
CONSTANT SubjectNo = 3
DECLARE StudentIndex:INTEGER
DECLARE TotalMark:REAL
DECLARE SubjectIndex:INTEGER
DECLARE StudentMark: ARRAY[1:ClassSize, 1:SubjectNo] OF REAL
DECLARE AverageMark:REAL
DECLARE StudentName: ARRAY[1:ClassSize] OF STRING
DECLARE DistinctionCount:INTEGER
DECLARE MeritCount:INTEGER
DECLARE PassCount:INTEGER
DECLARE FailCount:INTEGER

DistinctionCount <- 0
```

```
MeritCount <- 0
PassCount <- 0
FailCount <- 0

// Library Routine Declaration
FUNCTION pow(num:REAL, p:INTEGER) RETURNS REAL
    IF p = 0 THEN
        RETURN 1
    ENDIF
    RETURN num*pow(num, p-1)
ENDFUNCTION
FUNCTION ROUND(num:REAL, dp:INTEGER) RETURNS REAL
    DECLARE down:REAL
    down <- ((num*pow(10, dp)) DIV 1) / pow(10, dp)
    IF num-down >= 0.5/pow(10, dp) THEN
        RETURN down + 1/pow(10, dp)
    ELSE
        RETURN down
    ENDIF
ENDFUNCTION
FUNCTION MYRANDOM() RETURNS REAL
    RETURN RAND(1)
ENDFUNCTION


// Procedure to populate arrays
PROCEDURE PopulateArrays()
    DECLARE i:INTEGER
    DECLARE j:INTEGER
    FOR i <- 1 TO ClassSize
        OUTPUT "Enter name of student ", i
        INPUT StudentName[i]
        FOR j <- 1 TO SubjectNo
            StudentMark[i,j]  <- MYRANDOM()*100
        NEXT j
    NEXT i
ENDPROCEDURE
CALL PopulateArrays

// Loop through each student
```

```
FOR StudentIndex <- 1 TO ClassSize
    TotalMark <- 0

    // Loop through each subject for this student
    FOR SubjectIndex <- 1 TO SubjectNo
        TotalMark <- TotalMark + StudentMark[StudentIndex, SubjectIndex]
    NEXT SubjectIndex

    // Calculate average mark rounded to nearest whole number
    AverageMark <- ROUND(TotalMark / SubjectNo, 0)

    // Output the student’s name, total mark, and average mark
    OUTPUT "Student: ", StudentName[StudentIndex]
    OUTPUT "Total Marks: ", ROUND(TotalMark, 2)
    OUTPUT "Average Marks: ", AverageMark

    // Determine the grade based on the average mark and output it
    IF AverageMark >= 70 THEN
        OUTPUT "Grade: Distinction"
        DistinctionCount <- DistinctionCount + 1
    ENDIF
    IF AverageMark >= 55 THEN
        OUTPUT "Grade: Merit"
        MeritCount <- MeritCount + 1
    ENDIF
    IF AverageMark >= 40 THEN
        OUTPUT "Grade: Pass"
        PassCount <- PassCount + 1
    ENDIF
    IF AverageMark<40 THEN
        OUTPUT "Grade: Fail"
        FailCount <- FailCount + 1
    ENDIF
NEXT StudentIndex

// Output the overall class grade statistics
OUTPUT "Total Distinctions: ", DistinctionCount
OUTPUT "Total Merits: ", MeritCount
OUTPUT "Total Passes: ", PassCount
OUTPUT "Total Fails: ", FailCount
```