2. Look back at last week’s tasks. Describe the run-time bounds of these algorithms using the BigO notation.

The first task from last week has a Big O notations of O(n). This comes from the total efficiency calculation of 6n+4.

For every item in the list the ‘random\_arrange’ function will be run.

If the list has 0 items in it the efficiency drops to O(1) as the function is not run. This is the lower bound. As n tends to the infinite the efficiency tends towards infinite in a linear relationship.

The second function is for calculating the trailing 0’s of a decimal number. Its total efficiency is 4n+2 which gives it an O notation of O(n). When n is 0 the O notation is O(1) and as n increases towards infinite the efficiency tends towards the infinite in a linear relationship.

3. Write the pseudocode corresponding to functions for addition, subtraction and multiplication of two matrices, and then compute A = B\*C-2\*(B+C), where B and C are two matrices of order N. What is the run-time?

To create a matrix;

M <- INPUT matrix

FOR i <- 1 TO n

FOR j <- 1 TO n

INPUT element at position M[i][ j]

M[i][j]

To add two matrices;

// A and B are existing matrices, C is the matrix of A + B

MATRIX-ADDITION (A, B)

FOR i <- 1 TO n

FOR j <- 1 TO n

C[i][j] <- A[i][j] + B[i][j]

return C

To subtract one matrix from another;

// A and B are existing matrices, C is the matrix of A - B

MATRIX-ADDITION (A, B)

FOR i <- 1 TO n

FOR j <- 1 TO n

C[i][j] <- A[i][j] - B[i][j]

return C

To multiply two matrices;

// A and B are existing matrices, C is the matrix of A \* B

MATRIX\_MULTIPLICATION (A, B)

IF LENGTH(A) = LENGTH(B[i])

FOR i <- 1 TO n

ELSE

return FALSE