# Name: Jihyo Kim ID: A01017545 Set: 3M

# Assignment overview

The goals of this lab are:

* Practice evaluating code to determine its efficiency class
* Practice solving problems and writing pseudocode for your solutions

You may (and should!) discuss the lab and coding techniques with your classmates, but all of the work you submit to Learning Hub must be your own.

# Submission information

**Due date**: As shown on Learning Hub. Late assignments will not be graded.

What to submit:

* This Word document. Please type your answers in blue for visibility.

# Grading

This lab is worth 15 marks.

# Problem 1

Consider the following code:

Algorithm NOP (A[0..N-1])

for i = 0 to N-1 do

temp 🡨 A[N-1-i]

A[N-1-i] 🡨 A[i]

A[i] 🡨 temp

return A

1. (1 mark) What does this code do?  
     
   Reverse the order of elements in the list
2. (1 mark) How many times is the basic operation performed?  
   N times
3. (1 mark) What is the big-O efficiency class of this algorithm?  
   O(n)

# Problem 2

Data analysts often encounter missing or corrupted values in their data sets. One way to handle this is to substitute an estimated value for each bad item; this is called *imputation*. One method of doing imputation is called *mean substitution*: The average (arithmetic mean) of all the present/valid numbers is calculated and used to replace all of the bad numbers.

1. (4 marks) Write pseudocode for an algorithm to perform mean substitution on a data set. The data is stored in an array of length N. Negative numbers and zero are considered invalid data and are to be replaced by the arithmetic mean of the valid numbers.  
   MeanSubstitution (A[0..N-1])

count = 0

for i = 0 to N-1 do

if A[j] > 0

sum 🡨 sum + A[i]

count 🡨 count + 1

mean 🡨 sum / count

for j = 0 to N-1 do

if A[j] <= 0

B[j] 🡨mean

else

B[j] 🡨 A[j]

return B

1. (2 marks) What is the big-O efficiency class of your algorithm?  
   O(n)

# Problem 3

1. (4 marks) Write pseudocode for an algorithm that takes two integers as input:   
   N (>= 1) and B (>= 2), and returns an array of integers representing the digits of N in base B. For example, if N=37 and B=2, the resulting array would be [1,0,0,1,0,1]. If N=111 and B=7, the resulting array would be [2,1,6].  
     
   Assume that arrays are dynamic and magical, i.e. there is no need to declare your array to be of a particular size, or to increase storage as it grows. It will magically grow if you assign a value to an array element that has not been referenced before. However, there is a limit to the magic: You cannot insert an element into the beginning (or middle) of the array unless you shift over all the other elements. And finally, an array has a “length” property that tells the current size of the array. Arrays are indexed from 0 to length-1.  
     
   SomeAlgorithm(N, B)

counter 🡨0

while N > 0 do

remainder 🡨N % B

N 🡨N / B

A[counter] = remainder

counter 🡨counter + 1

for i = 0 to counter - 1 do

temp 🡨A[i]

A[i] 🡨 A[counter - i - 1]

A[counter - i -1] 🡨temp

return A

1. (2 marks) What is the big-O efficiency class of your algorithm?  
   O()