# Programming Guide for Introduction to Data Structures Assignment

## 1. Find Max and Min in a List

### Overview

This function finds the maximum and minimum values in a list of numbers.

### Pseudo Code

FUNCTION find\_max\_min(numbers)  
 IF numbers is empty  
 RETURN None, None  
  
 INITIALIZE max\_value to a very small number  
 INITIALIZE min\_value to a very large number  
  
 FOR each number in numbers  
 IF number is greater than max\_value  
 SET max\_value to number  
 IF number is less than min\_value  
 SET min\_value to number  
  
 RETURN max\_value, min\_value  
END FUNCTION

### Implementation Guide

* Start by checking if the list is empty. If it is, return a tuple of (None, None).
* Initialize two variables, max\_value and min\_value, to hold the maximum and minimum values found in the list.
* Iterate through each number in the list, updating max\_value and min\_value as you find larger or smaller numbers.
* Finally, return a tuple containing the max\_value and min\_value.

## 2. Check Symmetry of a String

### Overview

This function checks whether a given string is symmetrical (the same forwards and backwards).

### Pseudo Code

FUNCTION check\_symmetry(string)  
 SET start to 0  
 SET end to length of string - 1  
  
 WHILE start is less than or equal to end  
 IF character at start is not equal to character at end  
 RETURN False  
 INCREMENT start  
 DECREMENT end  
  
 RETURN True  
END FUNCTION

### Implementation Guide

* Compare characters at the beginning and the end of the string, moving inwards.
* If at any point the characters do not match, the string is not symmetrical, and you should return False.
* If you successfully compare all pairs of characters without finding a mismatch, return True.

## 3. Merge Sorted Lists

### Overview

This function merges two sorted lists into one single sorted list.

### Pseudo Code

FUNCTION merge\_sorted\_lists(list1, list2)  
 INITIALIZE merged\_list as an empty list  
 INITIALIZE index1 to 0  
 INITIALIZE index2 to 0  
  
 WHILE index1 is less than length of list1 AND index2 is less than length of list2  
 IF element at index1 of list1 is less than element at index2 of list2  
 APPEND element of list1 to merged\_list  
 INCREMENT index1  
 ELSE  
 APPEND element of list2 to merged\_list  
 INCREMENT index2  
  
 WHILE index1 is less than length of list1  
 APPEND remaining elements of list1 to merged\_list  
 INCREMENT index1  
  
 WHILE index2 is less than length of list2  
 APPEND remaining elements of list2 to merged\_list  
 INCREMENT index2  
  
 RETURN merged\_list  
END FUNCTION

### Implementation Guide

* Create an empty list merged\_list to store the merged elements.
* Use two indices, one for each list, to track your position as you iterate through the lists.
* Compare elements from both lists and append the smaller one to merged\_list.
* If one list is exhausted before the other, append the remaining elements of the other list to merged\_list.
* Return the merged\_list which now contains elements from both lists in a sorted manner.