Candidate Name	**Interview Date**
Interviewer	**Score**

Problem 01

A palindrome is a word or sequence of characters which reads the same backward or forward. For example the words: `level, anna, noon, rotator` are all palindromes. Write a function `palindrom` that accepts a string as an argument and returns a boolean indicating whether the input is a palindrome or not, for example:

```
palindrome("anna")   # returns True

palindrome("apple")  # returns False
```

Problem 02

Write the Linux command needed to change a file name from `original.txt` to `changed.txt`

Problem 03

Given a string containing characters (a-z), implement a function `runLengthEncode` that compresses repeated 'runs' of the same character by storing the length of that run, and provide a function `runLengthDecode` to reverse the compression. The output can be anything, as long as you can recreate the input with it.
For example:

```
runLengthEncode("aaaaaaaaaabbbaxxxxyyyzyx")   # returns "a10b3a1x4y3z1y1x1"

runLengthDecode("a10b3a1x4y3z1y1x1")          # returns "aaaaaaaaaabbbaxxxxyyyzyx"
```

Problem 04

Let `f` and `g` be two one-argument functions. The composition `f` after `g` is defined to be the function $x \mapsto f(g(x))$. Define a function `compose` that implements composition. For example, if `inc` is a function that adds `1` to its argument, and `square` is a function that squares its argument, then:

```
h = compose(square, inc)    # create a new function h by composing inc and square

h(6)                        # returns 49
```

Problem 05

Write a function `unique` that takes an array of strings as input and returns an array of the unique entries in the input, for example:

```
unique(['apples', 'oranges', 'flowers', 'apples'])    # returns ['oranges', 'flowers']
```

```
unique(['apples', 'apples'])                    # returns []
```

Problem 06

In linear algebra, the transpose of a matrix A is another matrix A^T created by writing the rows of A as the columns of A^T, for example:

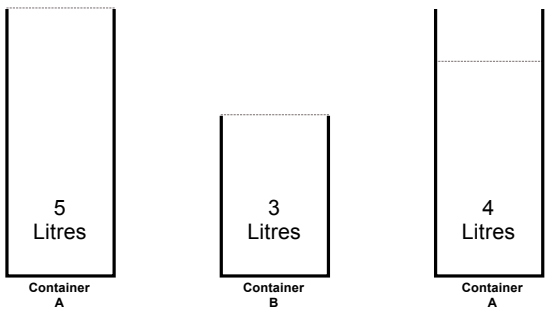
$$\begin{bmatrix} a_{11} & a_{12} \\ b_{21} & b_{22} \end{bmatrix}^T = \begin{bmatrix} a_{11} & b_{21} \\ a_{12} & b_{22} \end{bmatrix}$$

Write a function `transpose` that transposes a matrix, for example:

```
transpose( [ [1,2], [3,4] ] )          # returns  [ [1,3], [2,4] ]
transpose( [ [1,2,3,4], [5,6,7,8] ] )  # returns  [ [1,5], [2,6], [3,7], [4,8] ]
```

Problem 07

You are given 2 containers: `A` and `B`. Container `A` can hold `5 litres` of water, while container `B` can hold `3 litres`. You are also given a source of water that you can use as you wish. Show how you can use the containers and the water source to put exactly `4 litres` of water in container `A`. *No coding required, just write down the steps.*



Problem 08

Given an *integer* array of length `n`, find the index of the first duplicate element in the array and state the runtime and space complexity of your implementation, for example:

```
# returns 3, assuming the index starts with 0
index_of_first_duplicate( [ 5, 17, 3, 17, 4, -1 ] )
```

Problem 09

Given the below tree structure, write a function `sum` that accepts a `node` and returns the sum of integers for the whole tree represented by the given `node` argument

```
struct Node {
  value: Integer,
  children: [Node] # array of nodes
}
```