Scientific Computation
Autumn 2024
Problem sheet 1

---

1. Consider the Python function shown below:

```
def problem1(L):
    n = len(L)
    for i in range(n-1):
        for j in range(i+1,n):
            if L[i]==L[j]:
                return True
    return False
```

   (a) What is the problem that this code is attempting to solve?

   (b) Discuss the correctness of the code. Can it return `True` when it should return `False`? Is it possible for it to return `False` when it should return `True`?

   (c) Analyze the worst-case computational cost of the code. Use your analysis to find the asymptotic time complexity of the underlying algorithm (expressed using big-O notation).

2. The ith iteration of insertion sort requires the determination of the correct location of the $i+1$th element in a list. Assume that this location can be found arbitrarily quickly. What then is the worst-case big-O cost of the insertion sort? (The time complexity of operations using lists and other Python containers can be found here: https://wiki.python.org/moin/TimeComplexity)

3. In lecture 3, we defined a family of hash functions for IP addresses: $H(a) = \sum_{j=1}^{4} w_j a_j$ rem p where p is a prime that has been specified, and each weight, w_j, is chosen uniformly at random from $\{0, 1, 2, ..., p-1\}$. Given two distinct IP addresses, how many hash functions in this family will assign the same index to these addresses? Say we have two hash functions, $H^{(1)}$ and $H^{(2)}$, and two IP addresses, x and y. We are interested in cases where $H^{(i)}(x) = H^{(i)}(y)$, but we do not require $H^{(1)}(x) = H^{(2)}(x)$.