FIT2004: Lab questions for week 3

Objectives: This prac allows you to explore the concepts learnt in week 1-3.
NOTE: This prac is **NOT** assessed. You are required to write all programs in Python.

1. Implement the insertion sort algorithm.
2. Implement the binary search algorithm. Since it is very easy to get the details of binary search wrong when coding it so it is a good idea to test it on at least the following cases:
   (a) an empty array!
   (b) an array of one element,
      i. equal to the key or
      ii. less than it or
      iii. greater than it
   (c) several elements with the key less than the 1st or
   (d) the key greater than the last element
   (e) the key equal to the 1st, a middle, or the last element
   (f) the key not in the array and falling between various positions
3. Let `A[...]` be an array of N floating point numbers; N might be very large indeed. `A[...]` contains some (volatile) floating-point data. We want to write a method to "smooth" this data.

   (i) Write efficient program to print a smoothed version of the data in `A[...]`, smoothing over a sliding "window" of width w, where w is an odd, positive integer. Let the smoothed array be called Smoothed. `Smoothed[i]` will contain the average of all the elements in A[j ... k] where $j = i - floor(w/2)$ and $k = i + floor(w/2)$. If $j < 0$, we will use $A[0]$ and if $k \geq N$, we will use `A[N-1]` (the last element of the array). Below is an example.

   If $w = 3$, `Smoothed[0]= 3.0` which is the average of `A[0], A[0], A[1]` (i.e., `[2.0,2.0,5.0]`). Similarly, `Smoothed[1]= 4.0` which is the average of [2.0 5.0 5.0]. `Smoothed[6] = 4.0` which the average of `A[4], A[5], A[5]` (i.e., $[2.0, 5.0, 5.0]$).

```
For w = 3:
          A[]=   2.0 5.0 5.0 8.0 2.0 2.0 5.0
smoothed A[]=   3.0 4.0 6.0 5.0 4.0 3.0 4.0
```

A straightforward implementation takes $O(wN)$ time. Your algorithm should run in $O(N)$ time.

(ii) Give a logical argument why your solution to part (i)

a) terminates b) correct.

4. Implement Quick Select algorithm covered in week 3. You may reuse the code for Quick Sort shown in week 3 lecture.

```
-=0=-
 END
-=0=-
```