Practical 6

1 Question 1

Consider the numbers K_n defined by

$$K_n = \begin{cases} n & \text{for } n \leq 3 \\ K_{n-1} + 2K_{n-2} + 3K_{n-3} & \text{for } n \geq 4 \end{cases}$$

(a) Calculate K_8

- $K_8 = K_7 + 2k_6 + 3k_5 = 293$
- $K_7 = k_6 + k_5 + 2k_4 = 125$
- $k_6 = k_5 + 2k_4 + 3k_3 = 22 + 20 + 9 = 51$
- $k_5 = k_4 + 6 + 6 = 22$
- $k_4 = 3 + 4 + 3 = 10$

(b) Write pseudocode for a recursive function that returns K_n for an integer $n > 0$

```
def K (n)
if n ⩽ 3 then
    return n
else
    return K(n-1)+2K(n-2)+3K(n-3)
```

(c) Write pseudocode for a non recursive function that returns K_n for an integer $n > 0$

```
list L[1,2,3]
input n
for i in range 3,n do
    L.append(i[n-1]+2i[n-2]+3i[n-3])
end for
print L[n+3]
```

(d) Which of the two functions you have defined would it be better to implement

- Both would take a similar time and storage capacity as they both have to store the data in a list/stack for the results from 1 up to n, and so are performing a very similar operation

2 Question 2

Consider the problem of finding increasing subsequences in a sequence of integers. For example, in the sequence:

$$2, 6, 1, 9, 4, 8, 5, 10, 7$$

2,6,9 is such a subsequence, 1,4,8,10 is another and 1,4,5,7 a third (since the numbers appear within the sequence in that order, not necessarily consecutively). Use recursion to describe a procedure for finding the longest increasing subsequence. (Is using recursion the best approach?)

```
LIS(s)
result R
if len(S) =1
    return s
else
    x=s[-1]
    L=LIS(s-x)
    t=s with x and larger integers removed
    M=LIS(t)+x
    return whichever of L and M is longer
end if
```

3 Question 4

```
Input: coordinates of the midpoint(mp), the length of the line (l) and recursion depth d.
Output: drawing of a H Tree.
    calculate the points of the H h1,h2,h3 and h4
if d = 0 then
    draw a H with midpoint mp and line length l
else
    draw a H on each of the points of the h1...h4 with line length l/2
end if
```