Recursion & Algorithms
Sample Sols

1. Recursion is a simple yet powerful concept in developing algorithms for solving complex problems. In programming it is the ability of a method to call itself. it is used in routines that solve problems by repeatedly processing the output of the same process – breaking a hard problem into smaller tasks.

2. a) $t(1) = 3$

$t(n) = t(n-1) + 4 \quad (n>1)$

$$t(2) = t(2-1) + 4$$
$$= t(1) + 4$$
$$= 3 + 4$$
$$t(2) = 7$$

$$t(3) = t(3-1) + 4$$
$$= t(2) + 4$$
$$= 7 + 4$$
$$t(3) = 11$$

$$t(4) = t(4-1) + 4$$
$$= t(3) + 4$$
$$= 11 + 4$$
$$t(4) = 15$$

b) $t(1) = 0$

$t(n) = 2t(n-1) + 1 \quad (n>1)$

$t(2) = 2t(2-1) + 1$

$= 2t(1) + 1$

$= 2(0) + 1$

$t(2) = 1$

$t(3) = 2t(3-1) + 1$

$= 2t(2) + 1$

$= 2(1) + 1$

$t(3) = 3$

$t(4) = 2t(4-1) + 1$

$= 2t(3) + 1$

$= 2(3) + 1$

$t(4) = 7$

3. a) non-Recursive. $2, 4, 6, 8, 10, 12, \ldots$

$t(n) = 2n$

e.g. when $n=1$ $t(1) = 2(1) = 2$

$n=2$ $t(2) = 2(2) = 4$

$n=3$ $t(3) = 2(3) = 6$ and so on.

Recursive

$t(1) = 2$

$t(n) = t(n-1) + 2 \quad (n>1)$

↖ contains the previous term/ defined in terms of itself!

b) 2, 5, 8, 11, 14, 17, ... Begin with 2 and go up in steps of 3.

Non-Recursive

$t(n) = 3n - 1$

e.g. $t(1) = 3(1) - 1$

$= 2$

$t(2) = 3(2) - 1$

$= 5$

$t(3) = 3(3) - 1$

$= 8$

and so on.

Recursive

$$\left.\begin{array}{l} t(1) = 2 \\ t(n) = t(n-1) + 3 \quad (n>1) \end{array}\right\}$$

e.g. $t(2) = t(2-1) + 3$

$= t(1) + 3$

$= 2 + 3$

$= 5$

$t(3) = t(2) + 3$

$= 5 + 3$

$= 8$

$t(4) = t(3) + 3$

$= 8 + 3$

$= 11$

and so on.

4. a)

```
1. Input m
2. t ← 2
3. output t
4. For n = 2 to m do
    4.1 t ← t + 2
    4.2 Output t
```

b)

```
1. Input m
2. t ← 2
3. Output t
4 For n = 2 to m do
    4.1 t ← t + 3
    4.2 output t
```