Problem 1

a) eval (a[0..n], x)

```
result ← a[0]
term ← 1
for i ∈ 1 to n do
    term ← term · x
    result ← result + a[i] · term
return result
```

b) $T(n) \in O(n)$ since the for loop executes $n+1$ times.
We can also say $T(n) \in \Theta(n)$ because there is no best case requiring fewer than $n+1$ iterations.
Recall $T(n) \in \Theta(n)$ iff $T(n) \in O(n)$ and $T(n) \in \Omega(n)$.

c) The algorithm takes $k+1$ steps, so it is $O(\log n)$.
It will compute x^n by squaring the intermediate value

```
eval2(a, b, n, x)
    while n > 1 do
        x ← x · x
        n ← n/2
    return a · x + b
```

d) The algorithm will be very similar, except we keep the value of original x and we check to see if n is odd:

```
eval3(a, b, n, x)
    returns term(n, x) · a + b

term(n, x)
    if n = 0 then return 1
    half ← term(⌊n/2⌋, x)
    if n is odd then
        return half · half · x
    else return half · half
```

The algorithm implements the recurrence

$$x^{2k+1} = (x^k)^2 \cdot x$$
$$x^{2k} = (x^k)^2.$$

Its running time is also $O(\log n)$ as we divide the exponent by two at each iteration

Marking scheme a) { 2 pt for clean i/o; 2 pts computing x^i; 2 pts for the sum.

b) { 2 pts for $O(\cdot)$; 2 pts for $\Theta(n)$

c) d) { 2 pts: explain $T(n)$; 1 pt: i/o clean; 2 pts: calculation

Prob. 2. a)

Quicksort has $O(n^2)$ efficiency in worst case.

ex: 5, 4, 3, 2, 1

Each partition splits the problem in one of size $n-1$ & 1.

b) We can use the median procedure to select the median as pivot. Since median is $O(n)$, this does not change the complexity of the conquer part, but now the partitions are guaranteed to be balanced.

$\Rightarrow T(n) \in O(n \log n)$ in worst case.

Prob 3. a

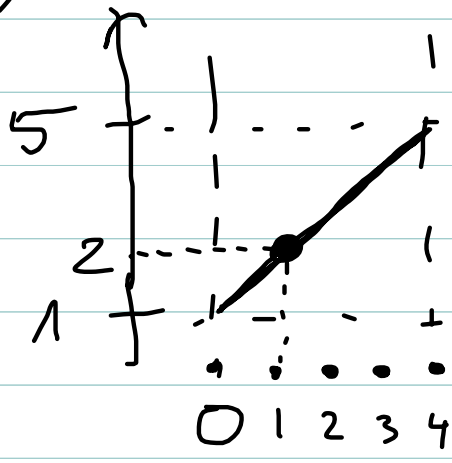
index: 0 1 [2] 3 4

↓

binary search.

$\Rightarrow i = 2.$

b)



$i = 1.$

Pr. 3, c

1, 1, 2, 3, 5

The search terminates since the key 2 is found after first comparison

2 pts : value
1 pt : explanation

d) 1, 3, 3, 4, 5 for example.

As long as $A[1] > 2$, then in interpolation search we know we should try an index smaller than 1. However, we know that $A[0]$ does not contain the key (2) because we would have chosen $i = 0$ as first index (linear interpolation). So, we could return "key not found" immediately after first comparison.

(4) Recurrence relations:

A) $T(n) = 5\,T(\frac{n}{2}) + n$

since $5 > 2^1$

$\Rightarrow T(n) \in O(n^{\log 5})$. $2 < \log 5$

B) $T(n) = 2\,T(n-1) + c$

$= 2(2T(n-2) + c) + c = 2^2 T(n-2) + (2+1)c$

$= 2^2(2T(n-3) + c) + (2+1)c = 2^3 T(n-3) + (2^2 + 2 + 1)c$

$\cdots \in O(2^n)$

c) $T(n) = 9T\left(\frac{n}{3}\right) + n^2$

since $9 = 3^2 \Rightarrow T(n) \in O(n^2 \log n)$

Algorithm C has best time complexity

Marking scheme

3 pts for each recurrence

1 pt for arguing for best algorithm.