Problem 1

1. $\#$ of comparisons $=$ the depth of the deepest node
$=$ height of the tree

The maximum no of leaves $= 2^h$

Implying the $\#$ of leaves, $L \leq 2^h$

$$\log L \leq \log 2^h$$

$$h \geq \lceil \log L \rceil \qquad L - \text{Number of permutations}$$

$$h \geq \lceil \log n! \rceil$$

$$n! = 4! = 24$$

$$h \geq \lceil \log 24 \rceil$$

$$h \geq \lceil 4.584 \rceil$$

$\#$ of comparisons $\geq 5$

Problem 2

2. $\{80, 27, 72, 1, 27, 8, 64, 34, 16\}$ Use Radix 9.

Since we are taking radix 9.

r[] =

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 16, 64, 27, 72, 27 | 1 | 34 | | | | | | 8, 80 |

q[] =

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 8, 1 | 16 | | 34, 27, 27 | | | | 64 | 80, 72 |

Result array $= \{1, 8, 16, 27, 27, 34, 64, 72, 80\}$

Problem 3

3) Algorithm occursOnce(s)

Input: A sequence of length $n$

Output: The first integer that occurs once in the range $0$–$3n-1$

```
bucketArray ← array of length 3n
i ← 0
while (i<n) do
    x ← bucketArray[s[i]]
    bucketArray[s[i]] = x+1

while (i<n) do
    x ← bucketArray[s[i]]
    if x = 1 then
        return x
```

The algorithm has $O(n)$ runtime because it goes through the bucket array twice at different times $\Theta(n)$ at both times making it $\Theta(2n)$. At the time of creating the bucketArray it also takes $\Theta(n)$ time.

$$\Theta(n) + \Theta(2n)$$

Asymptotic running time is $\Theta(n)$ "$\le$" $O(n)$

Problem 4

4) $f(0) = 0$, $f(1) = 1$, $f(n) = f(n-1) + f(n-2)$

a) Algorithm fib(n)
Input: integer n, $n \geq 0$
Output: The $n^{th}$ fibonacci number

```
if n = 0 || n = 1 then
    return 1
else
    return fib(n-1) + fib(n-2)
```

b) No you cannot

c) Algorithm fib(n)
Input: integer n, $n \geq 0$
Output: The $n^{th}$ fibonacci number

```
a ← 0
b ← 0
c ← 0
if n = 0 then
    return a
for i ← 2 to n-1
    c ← a+b
    a ← b
    b ← c
return b
```

d) The asymptotic runtime is $\Theta(n+c)$
$\Theta(n)$ or $O(n)$