Induction, Recursion, Algorithmic Analysis

Problem 1
Prove by induction that

$$1 \cdot 1! + 2 \cdot 2! + \ldots + n \cdot n! = (n+1)! - 1 \quad \text{for } n \geq 1$$

Base $n=1$ $1 \cdot 1! = 2-1$ ✓

R: if $1 \cdot 1! + 2 \cdot 2! + \cdots k \cdot k! = (k+1)! - 1$

Problem 2
Let $\Sigma = \{1, 2, 3\}$.

B: $sum(\lambda) = 0$.

R: $sum(aw) = a + \Sigma(w)$

(a) Give a recursive definition for the function $\mathsf{sum} : \Sigma^* \to \mathbb{N}$ which, when given a word over Σ returns the sum of the digits. For example $\mathsf{sum}(1232) = 8$, $\mathsf{sum}(222) = 6$, and $\mathsf{sum}(1) = 1$. You should assume $\mathsf{sum}(\lambda) = 0$.

(b) For $w \in \Sigma^*$, let $P(w)$ be the proposition that for all words $v \in \Sigma^*$, $\mathsf{sum}(wv) = \mathsf{sum}(w) + \mathsf{sum}(v)$. Prove that $P(w)$ holds for all $w \in \Sigma^*$.

(c) Consder the function $\mathsf{rev} : \Sigma^* \to \Sigma^*$ defined recursively as follows:

- $\mathsf{rev}(\lambda) = \lambda$
- For $w \in \Sigma^*$ and $a \in \Sigma$, $\mathsf{rev}(aw) = \mathsf{rev}(w)a$

Prove that for all words $w \in \Sigma^*$, $\mathsf{sum}(\mathsf{rev}(w)) = \mathsf{sum}(w)$

Problem 3
Define $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ recursively as follows: $f(m, 0) = 0$ for all $m \in \mathbb{N}$ and $f(m, n+1) = m + f(m, n)$.

(a) Let $P(n)$ be the proposition that $f(0, n) = f(n, 0)$. Prove that $P(n)$ holds for all $n \in \mathbb{N}$.

*(b) Let $Q(m)$ be the proposition $\forall n$, $f(m, n) = f(n, m)$. Prove that $Q(m)$ holds for all $m \in \mathbb{N}$.

Problem 4
Analyse the complexity of the following algorithms to compute the n-th Fibonacci number

(a) **FibOne**(n):

    if $n \leq 2$ then return 1
    else return **FibOne**$(n-1)$ + **FibOne**$(n-2)$

(b) **FibTwo**(n):

    $x = 1$, $y = 0$, $i = 1$
    While $i < n$:
        $t = x$

$x = x + y$

$y = t$

$i = i + 1$

return x

Problem 5

Analyse the complexity of the following recursive algorithm to test whether a number x occurs in an *ordered* list $L = [x_1, x_2, \ldots, x_n]$ of size n. Take the cost to be the number of list element comparison operations.

BinarySearch$(x, L = [x_1, x_2, \ldots, x_n])$:

if $n = 0$ then return no

else

if $x_{\lceil \frac{n}{2} \rceil} > x$ then return **BinarySearch**$(x, [x_1, \ldots, x_{\lceil \frac{n}{2} \rceil - 1}])$

else if $x_{\lceil \frac{n}{2} \rceil} < x$ return **BinarySearch**$(x, [x_{\lceil \frac{n}{2} \rceil + 1}, \ldots, x_n])$

else return yes