Inductive definitions of the factorial function [40 pts]

The goal in this problem is to prove the correctness of the following function $f$ which invokes a tail recursive function fact to calculate the factorial of a given integer:

$$
\begin{align*}
\text{fun} & \text{ fact } 0 \ a = a \\
| & \text{ fact } n \ a = \text{ fact } (n - 1) \ (n \times a) \\
\text{fun} & \ f \ n = \text{ fact } n \ 1
\end{align*}
$$

To prove the correctness of $f$, we rewrite the definition of fact into the following judgment $\text{fact } n \ a \ s$ and two inference rules:

$$\begin{align*}
\text{fact } n \ a \ s & \iff \text{fact } n \ a \text{ evaluates to } s \\
\text{fact } 0 \ a \ a & \text{ Fbase} \\
\text{fact } (n - 1) \ (n \times a) \ s & \text{ Frec, } n > 0
\end{align*}
$$

Prove Theorem 0.1 shown below. We assume $n \geq 0$ and $0! = 1$.

You may introduce a lemma in your proof. If you choose to introduce a lemma, state the lemma and prove it separately.

**Theorem 0.1.** If $\text{fact } n \ 1 \ s$, then $s = n!$.

**Lemma 0.2 (Factorial').** Assume $n \geq 0$ and $0! = 1$. If $\text{fact } n \ a \ s$, then $s = n! \times a$.

**Proof.** By rule induction on the judgment $\text{fact } n \ a \ s$.

**Case** $\text{fact } 0 \ a \ a \ F\text{base}$ where $n = 0$:

- $0! \times a = 1 \times a = a$
- $s = a = 0! \times a$

**Case** $\text{fact } (n - 1) \ (n \times a) \ s \ F\text{rec}$

- $s = (n - 1)! \times (n \times a)$ by IH on $\text{fact } (n - 1) \ (n \times a) \ s$
- $s = n! \times a$ from $(n - 1)! \times (n \times a) = n(n - 1)! \times a = n! \times a$

**Proof of Theorem 0.1.**

- $\text{fact } n \ 1 \ s$, then $s = n! \times 1 = n!$ by lemma 0.2.

**Untyped λ-calculus**

**Question 1.** [15 points] Complete the reduction rules for the call-by-name strategy. You may use the substitution $[c'/x]c$ as given in the course notes.
\[ e_1 \rightarrow e_1' \quad Lam \quad e_1 \mapsto e'/e' \quad App \]

**Question 2.** [15 pts] Show the reduction sequence of a given expression under the call-by-value strategy In each step, underline the redex. Do not rename bound variables.

\[ (\lambda x. \lambda f. f) \ (\lambda y. (\lambda z. z)) \ ((\lambda y'. y') \ (\lambda z'. z')) \]

\[ \mapsto \ (\lambda x. \lambda f. f) \ (\lambda z. z) \ ((\lambda y'. y') \ (\lambda z'. z')) \]

\[ \mapsto \ (\lambda f. f) \ ((\lambda y'. y') \ (\lambda z'. z')) \]

\[ \mapsto \ (\lambda f. f) \ (\lambda z'. z') \]

**Question 3.** [10 points] Fill in the blank with the set of free variables of the given expression.

\[
\begin{align*}
FV(\lambda x. x) &= \{\} \\
FV(x \ y) &= \{x, y\} \\
FV(\lambda x. x \ y) &= \{y\} \\
FV(\lambda x. \lambda y. x \ y) &= \{\} \\
FV((\lambda x. \lambda y. x \ y) \ (\lambda y. x \ y)) &= \{y, x\}
\end{align*}
\]

**Question 4.** [10 points] Give a redex whose reduction generates a variable capture.

\[ (\lambda x. \lambda y. x \ y) \ y \]

**Question 5.** [10 points] What is the result of \(\alpha\)-converting each expression in the left where a fresh variable to be generated in the conversion is provided in the right? Which expression is impossible to \(\alpha\)-convert?

\[
\begin{align*}
\lambda x. \lambda x'. x \ x' &\equiv_\alpha \lambda x'. \lambda x. x' \ x \\
\lambda x. \lambda x'. x \ x' \ x'' &\equiv_\alpha \lambda x'. \lambda x. x' \ x \ x'' \\
\lambda x. \lambda x'. x \ x' \ x'' &\equiv_\alpha \lambda x'' \text{impossible}
\end{align*}
\]