Name: ________________________________

Question 1 (20%)

Reduce the following expressions as much as you can – remember that application associates to left, $a\,b\,c$ means $((ab)c)$; and stacked lambdas associate to right, $(\lambda x \lambda y.x\,y\,y)$ means $(\lambda x.(\lambda y.x\,y\,y))$.

(a) $(\lambda z.z)(\lambda y.y\,y)(\lambda x.x\,a)$

$$\equiv_\beta (\lambda y.y\,y)(\lambda x.x\,a)$$
$$\equiv_\beta (\lambda x.x\,a)(\lambda x.x\,a)$$
$$\equiv_\beta (\lambda x.x\,a)a$$
$$\equiv_\beta a\,a$$

(b) $(\lambda x \lambda y.x\,y\,y)(\lambda a.a)b$

$$\equiv_\beta (\lambda y.(\lambda a.a)\;y\,y)b$$
$$\equiv_\beta (\lambda a.a)\;b\,b$$
$$\equiv_\beta b\,b$$

(c) $(\lambda x.x\,x)(\lambda y.y\,x)z$

$$\equiv_\beta (\lambda y.y\,x)(\lambda y.y\,x)z$$
$$\equiv_\beta (\lambda y.y\,x)xz$$
$$\equiv_\beta x\,x\,z$$

(d) $((\lambda x.x\,x)(\lambda y.y))(\lambda y.y)$

$$\equiv_\beta (\lambda y.y)(\lambda y.y)(\lambda y.y)$$
$$\equiv_\beta (\lambda y.y)(\lambda y.y)$$
$$\equiv_\beta (\lambda y.y)$$

Question 2 (45%)

Let the domain of entities $E = \{1,2,3,4,5\}$; and take the following denotations:

$[\![\text{woman}]\!] = \{1,2\}$

$[\![\text{man}]\!] = \{3\}$

$[\![\text{child}]\!] = \{5\}$

What would be the set-based interpretations of the following generalized quantifiers:

(a) Every woman and every man
$\{\{1,2,3\},\{1,2,3,4\},\{1,2,3,5\},\{1,2,3,4,5\}\}$

(b) Some woman but no man
$\{\{1\},\{2\},\{1,2\},\{1,4\},\{1,5\},\{2,4\},\{2,5\},\{1,2,4\},\{1,2,5\},\{1,4,5\},\{2,4,5\},\{1,2,4,5\}\}$

(c) Every man or some woman
$\{\{1\},\{2\},\{3\},\{1,3\},\{2,3\},\{3,4\},\{3,5\},\{1,2\},\{1,4\},\{1,5\},\{2,4\},\{2,5\},$
$\{1,2,3\},\{1,2,4\},\{1,2,5\},\{1,3,4\},\{1,3,5\},\{1,4,5\},\{2,3,4\},\{2,3,5\},\{2,4,5\},\{3,4,5\},$
$\{1,2,3,4\},\{2,3,4,5\},\{1,3,4,5\},\{1,2,3,5\},\{1,2,4,5\},\{1,2,3,4,5\}\}$

You can treat *but* exactly like *and*.

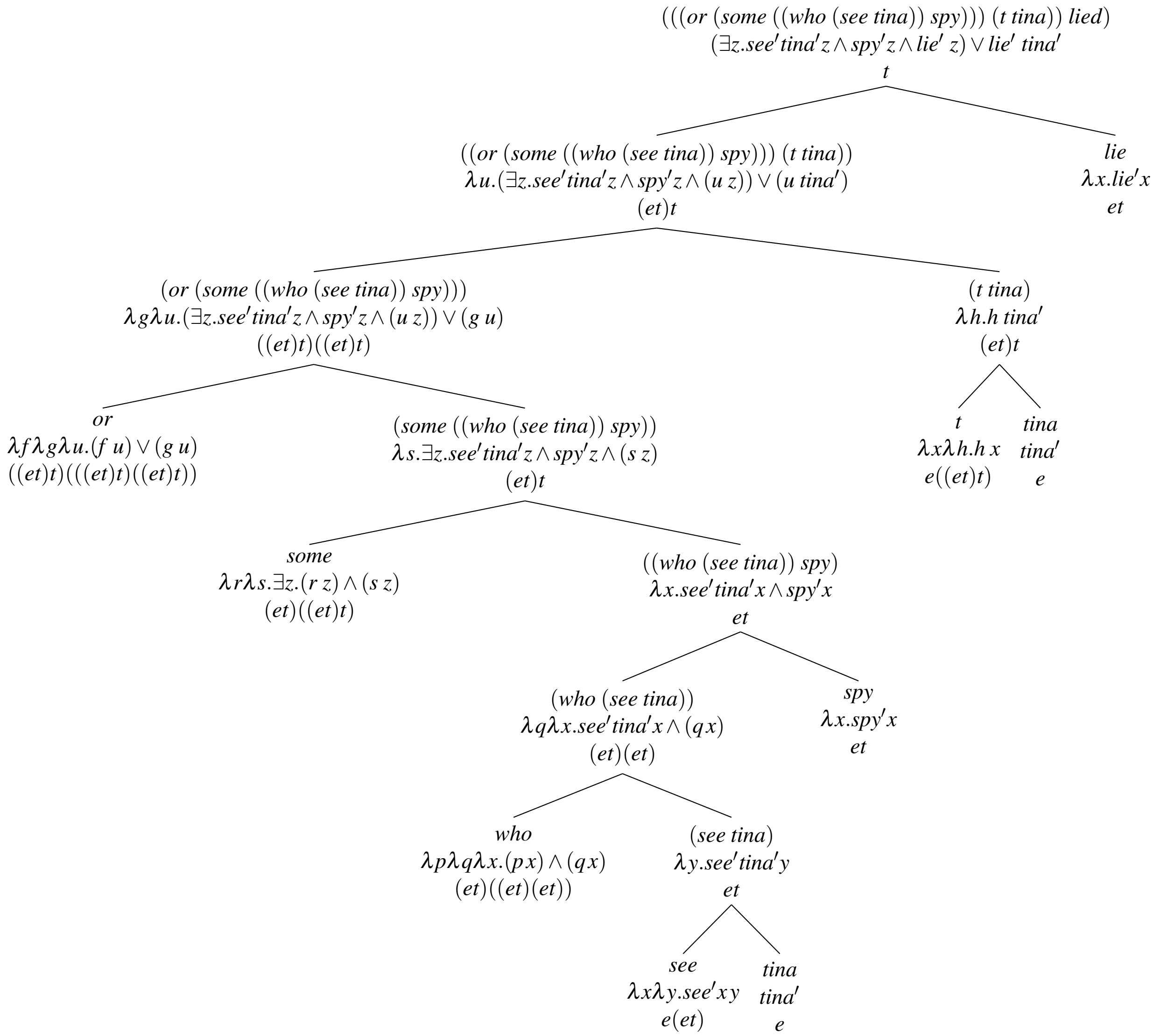
Question 3 (35%)

Derive the meaning of the following sentence on the basis of its applicative structure:

(1) a. Tina or some spy who saw Tina lied.
    b. $(((or\ (some\ ((who\ (see\ tina))\ spy)))\ (t\ tina))\ lie)$

You are required to draw a derivation tree, where each item has its type and lambda term shown. t is the operator that turns an e type individual interpretation to a generalized quantifier. You may give the interpretations in set notation or function notation, using '*' as the operator mapping one-place functions to corresponding sets. You can draw the parts of the tree separately to fit it into the page and/or draw it overleaf.

Logical notation:



Set-theoretic notation:

- $(((or\ (some\ ((who\ (see\ tina))\ spy)))\ (t\ tina))\ lied)$ — $lie'^{*} \in (\lambda s.(see'tina')^{*} \cap spy'^{*} \cap s^{*} \neq \emptyset)^{*} \cup (\lambda h.tina' \in h^{*})^{*}$ — t
  - $((or\ (some\ ((who\ (see\ tina))\ spy)))\ (t\ tina))$ — $\lambda x.x^{*} \in (\lambda s.(see'tina')^{*} \cap spy'^{*} \cap s^{*} \neq \emptyset)^{*} \cup (\lambda h.tina' \in h^{*})^{*}$ — $(et)t$
    - $(or\ (some\ ((who\ (see\ tina))\ spy)))$ — $\lambda g\lambda x.x^{*} \in (\lambda s.(see'tina')^{*} \cap spy'^{*} \cap s^{*} \neq \emptyset)^{*} \cup g^{*}$ — $((et)t)((et)t)$
      - or — $\lambda f\lambda g\lambda x.x^{*} \in f^{*} \cup g^{*}$ — $((et)t)(((et)t)((et)t))$
      - $(some\ ((who\ (see\ tina))\ spy))$ — $\lambda s.(see'tina')^{*} \cap spy'^{*} \cap s^{*} \neq \emptyset$ — $(et)t$
        - $some$ — $\lambda r\lambda s.r^{*} \cap s^{*} \neq \emptyset$ — $(et)((et)t)$
        - $((who\ (see\ tina))\ spy)$ — $(see'tina')^{*} \cap spy'^{*}$ — et
          - $(who\ (see\ tina))$ — $\lambda q.(see'tina')^{*} \cap q^{*}$ — $(et)(et)$
            - who — $\lambda p\lambda q.p^{*} \cap q^{*}$ — $(et)((et)(et))$
            - $(see\ tina)$ — $\lambda y.see'tina'y \equiv_{\eta} see'tina'$ — et
              - see — $\lambda x\lambda y.see'xy$ — $e(et)$
              - $tina$ — $tina'$ — e
          - spy — $\lambda x.spy'x \equiv_{\eta} spy'$ — et
    - $(t\ tina)$ — $\lambda h.tina' \in h^{*}$ — $(et)t$
      - t — $\lambda x\lambda h.x \in h^{*}$ — $e((et)t)$
      - $tina$ — $tina'$ — e
  - lie — $\lambda x.lie'x \equiv_{\eta} lie'$ — et