CSE 114A - Midterm fall 2021.

1A) There are multiple valid reder of the given expression. One of them is:

$(\lambda x\ y\ z \rightarrow x\ z\ y)\ (\lambda a\ b\ c \rightarrow a\ b\ c)$

This takes the form $(\lambda x \rightarrow e1)\ e2$

1B) Beta reduction to normal form

$=b\rangle$ $(\lambda y\ z \rightarrow (\lambda a\ b\ c \rightarrow a\ b\ c)\ z\ y)\ (\lambda x\ y \rightarrow y)$
$(\lambda y\ z \rightarrow y)$ apple

$=b\rangle$ $(\lambda z \rightarrow (\lambda a\ b\ c \rightarrow a\ b\ c)\ z\ (\lambda x\ y \rightarrow y))$
$(\lambda y\ z \rightarrow y)$ apple

$=b\rangle$ $(\lambda a\ b\ c \rightarrow a\ b\ c)\ (\lambda y\ z \rightarrow y)\ (\lambda x\ y \rightarrow y)$
apple

$=b\rangle$ $(\lambda b\ c \rightarrow (\lambda y\ z \rightarrow y)\ b\ c)\ (\lambda x\ y \rightarrow y)$ apple
$=b\rangle$ $(\lambda c \rightarrow (\lambda y\ z \rightarrow y)\ (\lambda x\ y \rightarrow y)\ c)$ apple
$=b\rangle$ $(\lambda y\ z \rightarrow y)\ (\lambda x\ y \rightarrow y)$ apple
$=b\rangle$ $(\lambda z \rightarrow (\lambda x\ y \rightarrow y))$ apple
$=b\rangle$ $\lambda x\ y \rightarrow y$

Answer is (d)

2A) All occurrences of x, y and z are bound in the expression.
'a' is free in the abstraction $(\lambda x \rightarrow x\ x\ a)$
'b' is free in the abstraction $(\lambda y \rightarrow y\ y\ b)$

Thus, the answer is (b)

2B) Beta reduction :

=b> $(\lambda y\ z \to y\ (\lambda x \to x\ x\ a)\ z)\ (\lambda y\ z \to z\ y)\ (\lambda y \to y\ y\ b)$

=b> $(\lambda z \to (\lambda y\ z \to z\ y)\ (\lambda x \to x\ x\ a)\ z)\ (\lambda y \to y\ y\ b)$

=b> $(\lambda y\ z \to z\ y)\ (\lambda x \to x\ x\ a)\ (\lambda y \to y\ y\ b)$

=b> $(\lambda z \to z\ (\lambda x \to x\ x\ a))\ (\lambda y \to y\ y\ b)$

=b> $(\lambda y \to y\ y\ b)\ (\lambda x \to x\ x\ a)$

=b> $(\lambda x \to x\ x\ a)\ (\lambda x \to x\ x\ a)\ b$

=b> $((\lambda x \to x\ x\ a)\ (\lambda x \to x\ x\ a)\ a)\ b$

=b> similarly, if you keep reducing this, you will arrive at the expression:

$(\lambda x \to x\ x\ a)\ (\lambda x \to x\ x\ a)\ a\ a\ a\ a\ b$

The answer is (d)

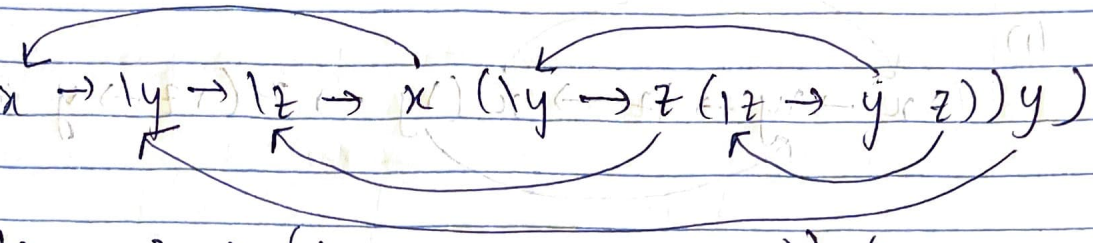
3) (A) ISZ n
(B) ONE
(C) MUL x (f x (DECR n))
(D) FIX POW2

Qn: To calculate 2^3 or POW (2,3)
Below are the recursive steps:

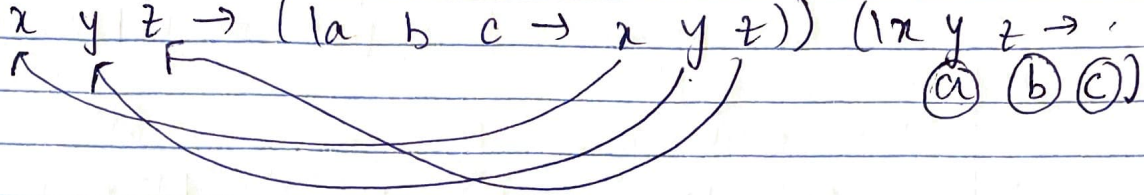
- 2 x pow (2,2)
- 2 x 2 x pow (2,1)
- 2 x 2 x 2 x pow (2,0) ← base case
- 2 x 2 x 2 x 1.
- = 8

4)

A) (\x → \y → \z → x (\y → z (\z → y z)) y)



B) (\x y z → (\a b c → x y z)) (\x y z → x y z)



5] map (\x y → x + y) [1, 2, 3, 4, 5]

The function (\x y → x + y) would be applied to every element of the list. A list of functions would be returned like:

[(\y → 1 + y), (\y → 2 + y), ...]

So, (d) is the correct answer.

Note that the function (\x y → x + y) will be partially applied to every list element

6) foldl (-) 0 [1, 2, 3]. Ans: (d)

foldl applies the function, (-) in this case to the accumulator and head of the list. It repeats this, while updating the accumulator, till the list is empty.

Accumulator states:

0 - 1 = -1

-1 - 2 = -3

-3 - 3 = -6 (at this point list is empty and -6 is returned). Ans is (d)

7) foldr (flip (-)) 0 [1, 2, 3]

foldr uses head recursion and extracts the head of the list, applies the constructor, flip(-) in this case, and recursively calls foldr on rest of the list.

Without flip (-), the execution of foldr would have been:

(1 - (2 - (3 - 0)))

but since the operation is flip (-), the execution is:

((0 - 3) - 2) - 1) = -6

Answer is (d)

8) (A) Abs x e
(B) Show e1 ++ " " ++ show e2.
(C) Var x = x

9) Abs "stp"
(App
(Abs "x" (App (Var "stp")
(App Var "x" Var "x")))
(Abs "x" (App (Var "stp")
(App Var "x" Var "x")))
)

10) (A) `if x == y then e else Var x`

this is because

$x\,[x := e] = e$, and

$y\,[x := e] = y$

- This can be read as "substitute all free occurrences of x by e in the expression x"
- Since there are no free occurrences of x in the expression y, simply y would be returned without any substitutions.

(B) `App (subst e1 y e3) (subst e2 y e3)`

This is because

$(e1\ e2)\,[x := e] = (e1\,[x := e])\ (e2\,[x := e])$

(C) `Abs x e1`

This is because

$(\backslash x \rightarrow e1)\,[x := e] = \backslash x \rightarrow e1$

Since there are no free occurrences of x in the expression $(\backslash x \rightarrow e1)$ - All occurrences of x are bound by $\backslash x$

(D) `Abs x (subst e1 y e2)`

This is the case where the free variable of e2 will NOT be captured by 'x' represented by 'notElem x (fv e2)'

11) The patterns are non-exhaustive. One possible call that will not match any of the patterns is :

subst (Abs "a" Var "a") "b" (Var "a")

Here, $x \neq y$ and the free variables of e2 (ie, a) will be captured by the abstraction

That is, notElem x (fv e2) is false. because x is present in the fv list of e2.