1. Given, $L = (\lambda x.\ y(xx))\ (\lambda y.x(yy))\ (\lambda z.y)$
   Say L = MNO
   where,
   $M = (\lambda x.\ y(xx))$
   $N = (\lambda y.\ x(yy))$
   $O = (\lambda z.y)$

   FreeVariable (M) = $FV[\lambda x.\ y(xx)]$
   $= FV\ [y(xx)] \setminus \{x\}$ ..(using $FV\ [(\lambda xM)] = FV\ (M) \setminus \{x\}$)
   $= \{x,y\} \setminus \{x\}$
   $= \{x\}$ **(1)**
   Similarly, $FV(N) = \{y\}$ **(2)**
   $FV(O) = FV\ (\lambda z.y)$
   $= \{y\}$ **(3)**

   Using left associativity, L = (MN)O
   So, $FV(L) = FV((MN)O)$
   $= FV(MN) \cup FV(O)$
   $= FV(M) \cup FV(N) \cup FV(O)$
   $= \{y\} \cup \{x\} \cup \{y\}$
   $= \{x,y\}$

2.
   (a) $(\lambda\ ab\ .\ ba)\ ab$
   α renaming => $(\lambda\ cd.\ dc)\ ab$
   β reduction => $dc[\ d:= a,\ c:= b]$
   => ba

   (b) $(\lambda\ x.\ xx)\ (\lambda\ a.\ a)$
   β reduction => $xx\ [\ x:= (\lambda\ a.\ a)]$
   => $(\lambda\ a.\ a)\ (\lambda\ a.\ a)$
   α renaming => $(\lambda\ a.\ a)\ (\lambda\ b.\ b)$
   β reduction => $a\ [a:= (\lambda\ b.\ b)]$
   => $(\lambda\ b.\ b)$

   (c) $(\lambda\ x.\ xx)\ (\lambda\ x.\ xx)$
   α renaming => $(\lambda\ x.\ xx)\ (\lambda\ y.\ yy)$
   β reduction => $xx[x:= (\lambda\ y.\ yy)]$
   => $(\lambda\ x.\ xx)\ (\lambda\ x.\ xx)$

3. $(\lambda\ x.\ xx)\ (\lambda\ x.\ xx)$
   As we have seen in 2(c), after doing α and β reductions, we again reach to that point from where we started. So it will be infinite loop and everytime it can be beta reduced further.

It is also justified as lambda calculus is equivalent to turing machines and it has infinite loops (I.e. non halting) and it is also here in form of such terms which can always be reduced.

4. So, our task is to encode (or x y)
   We can write it as:
   or x y = if (x == True) then True
   else False
   which can be stated as:
   if x then True else y

   This can be encoded using *if else then* encoding as discussed in lectures,

   if *a* then *b* else *c* = λ abc . abc

   Similarly, if x then True else y = λ xy . (x True y)

5. Using, fixed point of g = Yg
   Fixed point of (λ x. x) = Y (λ x. x)
   (λ k.(λ x.(k(xx)) (λ x.(k(xx))) (λ x. x)

   Now we will reduce it,
   α renaming => (λ k.(λ y. k(yy)) (λ z. k(zz))) (λ x. x)
   β reduction => (λ k.(λ y. k(yy)) (λ z. k(zz))) [k := (λ x. x)]
   => (λ y. (λ x. x) (yy)) (λ z. (λ x. x) (zz))

   Now we will simplify the inner terms, I.e. (λ x. x) (yy) and (λ x. x) (zz)
   This will give
   => (λ y.(x [x :=yy])) (λ z.(x [x := zz]))
   => (λ y. yy)(λ z. zz)

6. Given, sum = λ n. if n==0 then 0 else n+(sum (n-1))
   Now we need to define recursive form of sum,

   sum1 = λ f n . if n==0 then 0 else n+(f (n-1))

   Y sum1 will be the required solution.