1. For each type, construct a simply-typed lambda calculus term (variables, functions, and function application only) whose most general type is that type, or argue that no term has that type.
   (Hint: You can double-check your answers in OCaml. Extra credit: for any type that has no simply-typed lambda calculus term, give an OCaml term that does have the type.)
   (a) $\alpha \rightarrow \beta \rightarrow \beta$
   (b) $(\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \beta \rightarrow \alpha \rightarrow \gamma$
   (c) $\alpha \rightarrow \beta$
   (d) $\alpha \rightarrow \alpha \rightarrow \alpha$

2. Does the simply-typed lambda calculus with integers have a subject expansion property, meaning if $\Gamma \vdash e : \tau$ and $e' \rightarrow e$, does $\Gamma \vdash e' : \tau$? Here $\rightarrow$ is reduction under call-by-value semantics. Either prove that subject expansion holds, or give a counterexample showing that it does not hold.

3. Suppose we were to add booleans to the simply-typed lambda calculus:
   
   $$e ::= x \mid n \mid true \mid false \mid \lambda x.e \mid e e \mid if \ e \ then \ e \ else \ e$$
   
   (a) Write down small-step call-by-value semantic rules for the new forms $true$, $false$, and $if$. (Here $if$ should behave as it does in O’Caml, evaluating to the result of either the true or false branch depending on the guard.)
   (b) Extend the typing judgment $\Gamma \vdash e : \tau$ to the new forms $true$, $false$, and $if$.
   (c) Prove progress and preservation for the extended language. (You don’t need to reprove the cases for the old forms; make your arguments as extensions to the proof given in the lecture slides.)

4. Suppose we were to add the $Y$ combinator to the simply-typed lambda calculus for defining recursive functions:

   $$e ::= ... \mid Y$$

   As for the above, write small-step, call-by-value semantic rules for this combinator and its typing rule.
   
   Hint: since you will be defining call-by-value semantics, the $Y$ combinator should take a function as its argument—see the factorial example in the lecture notes.
   
   Extra credit: add appropriate cases to your progress and preservation lemmas to include $Y$ as well.