CS 496 QUIZ 7

Name: Cindy Zhang

Pledge: I pledge my honor that I have abided by the Stevens Honors System.

Exercise One. The following expression e is typable. Exhibit a typing environment Γ, a type t and a typing derivation for the judgement $\Gamma \vdash e::t$.

```
proc (x:int) { proc (y:int) { x-y } } has type int -> int -> int

x:int ⊢ proc (y:int) { x-y }          has type int -> int

x:int, y:int ⊢ { x-y }                has type int
```

```
Let's [x:int, y:int] be defined as:

Γ⊢ y:int                              TVar

Γ⊢ x:int                              TVar


Let's [Γ  x – y] be defined as:

Γ⊢ x:int                              TVar

Γ⊢ y:int                              TVar
```

Exercise Two. Give typable expressions of each of the following types, justifying your result by showing a type derivation fo each of them.

1. `bool -> int`

   `proc (y:bool) { if y then 1 else 2}`

   `y:bool ⊢ if y then 1 else 2`

   `Let [y:bool] be defined as:`

   `Γ⊢ y:bool`   `TVar`

   `Let [if y then 1 else 2] be defined as:`

   `Γ⊢ y`   `TVar`

   `Γ⊢ 1`   `TConst`

   `Γ⊢ 2`   `TConst`

2. `(bool -> int) -> int`

   `proc (y:bool) { proc (x:int) { if y then x else x+5} }`

   `Let's [y:bool, x:int] be defined as:`

   `Γ⊢ y:bool`   `TVar`

   `Γ⊢ x:int`   `TVar`

   `Let's [Γ if y then x else x+5] be defined as:`

   `Γ⊢ y`   `TVar`

   `Γ⊢ x`   `TVar`

   `Γ⊢ x+5`   `TVar`

3. bool -> (bool -> bool)

   proc (y:bool) { proc (x:bool) { if x then true else false } }

   Let's [y:bool, x:bool] be define as:

   Γ⊢ y:bool   TVar

   Γ⊢ x:bool   TVar

   Let's [if x then true else false]

   Γ⊢ x   TVar

   Γ⊢ true   TConst

   Γ⊢ false   TConst