26/30

```
// HW5
//Christopher Choitz


// This sets F# to read from whatever directory contains this source file.
System.Environment.set_CurrentDirectory __SOURCE_DIRECTORY__;;

#load "parser.fsx"

// This lets us refer to "Parser.Parse.parsefile" simply as "parsefile",
// and to constructors like "Parser.Parse.APP" simply as "APP".
open Parser.Parse



//Substituion utility function that the PCF uses.  Textual substitution is being used in
//this case to find the free identifier.
let rec subst = function
    | IF (e1, e2, e3) x t -> IF ((subst e1 x t), (subst e2 x t), (subst e3 x t))
    | APP (e1, e2) x t -> APP ((subst e1 x t), (subst e2 x t))
    | ID (v) x t -> if v=x then t else ID v
    | FUN (v, e) x t -> FUN (v, if v=x then e else (subst e x t))
    | REC (v, e) x t -> REC (v, if v=x then e else (subst e x t))
    | e _ _ -> e



//Interp function.  The skeleton program was provided by Prof. Smith.


let rec interp = function
| ID _ -> ERROR (sprintf "unbounded identifier")
| REC (x, e)    -> interp (subst e x REC (x, e))   //Rule (10)
| IF (e1, e2, e3) ->
    match (interp e1) with
    | (ERROR s) -> ERROR s
    | (BOOL true) -> interp e2   //Rule (4)
    | (BOOL false) -> interp e3  //Rule (5)
    |(_) -> ERROR (sprintf "if condition must be a boolean")
| APP (e1, e2) ->
    match (interp e1, interp e2) with
    | (ERROR s, _)  -> ERROR s        // ERRORs are propagated
    | (_, ERROR s)  -> ERROR s
    | (SUCC, NUM n) -> NUM (n+1)      // Rule (6)
```

6

20

Does not compile!
Syntax error. (only one pattern is allowed.)
But logic is good.

```
    | (SUCC, v)     -> ERROR (sprintf "'succ' needs int argument, not '%A'" v)
    | (PRED, NUM 0) -> NUM 0      //Rule (7)
    | (PRED, NUM n) -> NUM (n-1)
    | (PRED, v)     -> ERROR (sprintf "'pred' needs int argument, not '%A'" v)
    | (ISZERO, NUM 0) -> BOOL true  //Rule (8)
    | (ISZERO, NUM _) -> BOOL false
    | (ISZERO, v)   -> ERROR (sprintf "'iszero' needs int argument, not '%A'" v)
    | (FUN (x, e), v) -> interp (subst e x v)   //Rule (9)
    | (_, _)    -> ERROR (sprintf "not a funtional application")
| e -> e

// Here are two convenient abbreviations for using your interpreter.
let interpfile filename = filename |> parsefile |> interp

let interpstr sourcecode = sourcecode |> parsestr |> interp
```