//**Name: Aqib Shah (3353447), Michael Martinez (3382106)**

//Date: April 20th, 2015

//COP4555 - Assignment 5

System.Environment.set\_CurrentDirectory \_\_SOURCE\_DIRECTORY\_\_;;

#load "parser.fsx";;

open Parser.Parse;;

//subst function

let rec subst e x t =

match e with

| ID n ->

if n = x then t

else ID n

| NUM n -> NUM n

| BOOL bool -> BOOL bool

| SUCC -> SUCC

| PRED -> PRED

| ISZERO -> ISZERO

| IF (bool, e1, e2) -> IF (subst bool x t, subst e1 x t, subst e2 x t)

| APP (e1, e2) -> APP (subst e1 x t, subst e2 x t)

| FUN (e1, e2) ->

if x = e1 then FUN (e1, e2)

else FUN (e1, subst e2 x t)

| REC (e1, e2) ->

if x = e1 then REC (e1, e2)

else REC (e1, subst e2 x t)

| \_ -> ERROR (sprintf "error on subst call");;

//interp function

let rec interp = function

| NUM n -> //rule 1

match n with

| n when n < 0 -> ERROR (sprintf "'%A' is not a non-negative number" n)

| n -> NUM n

| BOOL bool -> //rule 2

match bool with

| true -> BOOL true

| false -> BOOL false

| SUCC -> SUCC //rule 3

| PRED -> PRED

| ISZERO -> ISZERO

| IF (bool, e1, e2) ->

match (interp bool, e1, e2) with

| (ERROR s, \_, \_) -> ERROR s // ERRORs are propagated

| (\_, ERROR s, \_) -> ERROR s

| (\_, \_, ERROR s) -> ERROR s

| (BOOL true, e1, e2) -> interp e1 //rule 4

| (BOOL false, e1, e2) -> interp e2 //rule 5

| (e1, e2, e3) -> ERROR (sprintf "Expected BOOL\*TERM\*TERM, but received %A, %A, %A" e1 e2 e3)

| 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

| (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 n) -> BOOL false

| (ISZERO, v) -> ERROR (sprintf "'iszero' needs int argument, not '%A'" v)

| (FUN (e1, e2), v) -> interp (subst e2 e1 v) //rule 10

| (REC (e1, e2), v) -> interp (APP(subst e2 e1 (REC (e1, e2)), v)) //rule 11

| (e1, e2) -> APP (e1, e2)

| ID x -> ID x

| FUN (x, e) -> FUN (x, e) //rule 9

| REC (x, e) -> REC (x, e) //rule 11

| ERROR s -> ERROR s;;

// Here are two convenient abbreviations for using your interpreter.

let interpfile filename = filename |> parsefile |> interp;;

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

interpfile "twice.pcf";;

//val it : term = NUM 65536

interpfile "double.pcf";;

//val it : term = NUM 74

interpfile "minus.pcf";;

//val it : term = NUM 46

interpfile "fibonacci.pcf";;

//val it : term = NUM 6765

interpfile "factorial.pcf";;

//val it : term = NUM 720

interpfile "divisor.pcf";;

//val it : term = NUM 191

interpfile "lists.pcf";;

//val it : term = BOOL true

interpfile "ackermann.pcf";;

//val it : term = NUM 509