```
(*PROBLEM 4*)
(*Pre-defined functions that were given/derived from the assignment*)
fun ifstat test stmt1 stmt2 =
 (fn x => if (test x) then (stmt1 x) else (stmt2 x))
;

fun seq stmt1 stmt2 =
 (fn x => (stmt2 (stmt1 x)))
;

fun assignx exp state =
 let val (x,y) = state
 in ((exp state),y)
 end
;

fun assigny exp state =
 let val (x,y) = state
 in (x, (exp state))
 end
;

(*Used in a seperate case where the state is of the form (x, y, z)*)
fun assignx3 exp state =
 let val (x,y,z) = state
 in ((exp state),y,z)
 end
;

fun assigny3 exp state =
 let val (x,y,z) = state
 in (x, (exp state),z)
 end
;

(*Prob 4.1: Functions that simulate the while-do and the repeat-until statements
 from our language for structured programming.*)

fun whilestat test stmnt =
 (fn x => if (test x) then
  let val temp = stmnt x
  in
  whilestat test stmnt temp
  end
  else
  x)
;

fun repeatstat test stmnt =
 (fn x =>
  let val temp = stmnt x
  in
  if (test temp) then temp
```

```
  else (repeatstat test stmnt temp)
 end)
 ;

(*Prob 4.2: A factorial and exponential function written based on the encoding
  for statement forms described in this problem

  Pseudo code for factorial
  (x, y) => (inital value, accumulator)
  fun factorial numPair =
   whilestat (function: x > 0) (seq (assigny: x*y) (assignx: x-1)) numpair

  Pseudo code for exponential
  (x, y, z) => (accumulator, exponent, base value)
  fun factorial numPair =
   whilestat (function: y > 0) (seq (assignx: x*z) (assigny: y-1)) numpair
*)

fun factorial numPair =
 whilestat (fn (x,y) => x > 0) (seq (assigny (fn (x, y) => x*y))
  (assignx (fn (x, y) => x-1))) numPair;

fun exponential numPair =
 whilestat (fn (x,y,z) => y > 0) (seq (assignx3 (fn (x,y,z)  => x*z))
  (assigny3 (fn (x,y,z) => y-1))) numPair;



(*Prob 4.3: Running the encodings of the two imperative programs*)

factorial(7, 1);
(*Result: (0,5040)*)

factorial(5, 1);
(*Result: (0,120)*)

exponential(1, 3, 2);
(*Result: (8,1,2)*)

exponential(1, 5, 4);
(*Result: (1024,1,4)*)
```

PDF document made with CodePrint using Prism