For some of these experiments, it may be a good idea to use the `limit` or `ulimit` command in your shell to control the growth of your stack segment.

1. In your Foundations course, you wrote recursive factorial using `lambda` as

   \[
   (\lambda (f \ x) \ (f \ f \ x)) \\
   (\lambda (f a c t \ n) \\
   \quad (\text{if } (= 0 \ n) 1 \ (* \ n \ (f a c t \ fact \ (- \ n \ 1)))) \ ) \ ) \\
   5 \ )
   \]

   Test if this runs correctly using the Scheme interpreter.

2. In this class, we wrote factorial as

   \[
   (\lambda m \ (m \ m)) \\
   (\lambda f a c t \\
   (\lambda n \\
   \quad (\text{IF } (= 0 \ n) 1 \ (* \ n \ ((f a c t) \ (- \ n \ 1)))) \ ) \ ) \ ) \\
   5 \ )
   \]

   Does the above code execute correctly using Scheme?

3. The functional \( F \) for the factorial function was defined as

   \[
   (\lambda f a c t \\
   (\lambda n \ (\text{if } (= 0 \ n) 1 \ (* \ n \ (f a c t \ (- \ n \ 1)))))) \)
   \]

   Note that there is no \((f a c t \ f a c t)\) in the body, only a single \textit{fact}. We also defined the \textit{Y} operator such that \((\textit{Y} \ F)\) returns the factorial function. \textit{Y} was defined in class as

   \[
   (\lambda y \ (\lambda f \ ((y \ y) \ f))) \\
   (\lambda y \ (\lambda f \ ((y \ y) \ f))) \)
   \]

   (a) Does \(((\textit{Y} \ F) \ 5)\) compute the factorial of 5 correctly using Scheme? Explain why it does or does not.

   (b) If it does not, can you modify the definition of \textit{Y} (without modifying \( F \)) so that the problem is resolved?