The assignment is to do all the exercises in my tutorial. They are listed here for your convenience.****This assignment is due 3/24****.

EXERCISE I1: define the combinator S = lambda x.lambda y.lambda z. x z (y z). The notation for lambda x.E

is fun x -> E. This is Curried function: it's not a function that takes three arguments, but one that takes x,  
returns a function that takes y, which then returns a function that takes z. The body applies x to z; this  
returns a function that is applied to the result of applying y to z. If this is too confusing, skip to I2:

\*\*\*EXERCISE I2: figure out how to apply printfn to 1 and g(2,4)

correctly. And in general, the proper way to apply curried functions.

The hint is in the error message.

\*\*\* EXERCISE I3: fix

> let f y =

while y>0 do

printfn "%d" y

y <- y-1;;

\*\*\* EXERCISE I4: devise an experiment to verify that F# uses static as

opposed to dynamic scoping.

\*\*\* EXERCISE I5. Devise an experiment to test if, when an array is

passed to a function as an argument, whether a reference (pointer) is

passed, or is the entire array is copied.

\*\*\*EXERCISE C1 (type up in editor and compile). Modify the tostring

function so that expressions such as 3 + -4 is printed as (3 -

4). However, (3 + --4) should be printed as (3 + 4) and not (3 - -4).

\*\*\*EXERCISE C2 : Write a function to return the length (size) of a stack

(as defined in program).

\*\*\*EXERCISE C3: Write a function convert an F# list into a stack, without

reversing the elements.

You should type up all exercises in one file and submit the assignment.