The purpose of this assignment is to reinforce material from the PL tutorial.

Question 1. Convert the following two *MiniLang* expressions written in the concrete syntax to its corresponding first-order abstract syntax:

```
let days be 7 in 3 + days
'ab' ^ 'cd'
```

Question 2. Write the type derivation trees for the two expressions from question 1 according to the static semantic rules.

Question 3. Evaluate (reduce) the two expressions from question 1 according to the dynamic semantic rules.

Question 4. Convert two expressions from question 1 to its corresponding *higher-order* abstract syntax, where the abstract syntax of *MiniLang*'s `let` expression is now in higher-order form.

Question 5. Provide formal definitions of the functions *free*(e) and *bound(e)* that compute the set of free and bound variables, respectively, for an expression e over *MiniLang*'s higher-order abstract syntax.

Question 6. Formally define $[e'/x]e$ so that e' is substituted only for *free* (not bound) occurrences of x in *MiniLang* expression e (again over the higher-order abstract syntax of *MiniLang*).

Question 7. The `cadd` function, defined in the Twelf file, `evaluation.elf`, on line 5. What category of functions (lambda abstraction, pi abstraction, etc) does `cadd` belong to?

Question 8. What does "`cadd/z (s z)`" return?

Question 9. The `cadd` function is used to compute the addition of two non-negative integers. Write the Twelf definition of the function `cmult` to compute the product of two non-negative integers.