4.1) //I sat up build and run file, so it is easy to run

a.

fsi Absyn.fs Fun.fs

```
open Absyn;;
open Fun;;
let res = run (Prim("+", CstI 5, CstI 7));;
#q;;
```

b.

run.bat

```
open Parse;;
let e1 = fromString "5+7";;
let e2 = fromString "let y = 7 in y + 2 end";;
let e3 = fromString "let f x = x + 7 in f 2 end";;
```

c.

run.bat

```
open ParseAndRun;;
run (fromString "5+7");;
run (fromString "let y = 7 in y + 2 end");;
run (fromString "let f x = x + 7 in f 2 end");;
```

4.2)

Write more example programs in the functional language, and test them in the same way as in Exercise 4.1:

- Compute the sum of the numbers from 1000 down to 1. Do this by defining a function sum n that computes the sum $n+(n-1)+\cdots+2+1$. (Use straightforward summation, no clever tricks).
  PLACED ANSWER IN PARSEANDRUN variable A42Run

- Compute the number 3^8, that is, 3 raised to the power 8. Again, use a recursive function.
  PLACED ANSWER IN PARSEANDRUN variable B42Run

- Compute $3^0 + 3^1 + \ldots + 3^{10} + 3^{11}$, using a recursive function (or two, if you prefer).
  PLACED ANSWER IN PARSEANDRUN variable C42Run

- Compute $1^8 + 2^8 + \cdots + 10^8$, again using a recursive function (or two).
  PLACED ANSWER IN PARSEANDRUN variable D42Run

4.3)

Extend to let Letfun and Call take argument list

- Done //but didn't know how to change FunPar.fsy to take a list with undefined length

4.4)

Extend FunPar.fsy to take any non-zero number of arguments

- NEED HELP

4.5)

- Changed: FunLex.fsl FunPar.fsy