Exercise 4.4

#

In continuation of Exercise 4.3, modify the parser specification to accept a language where functions may take any (non-zero) number of arguments. The resulting parser should permit function declarations such as these:

Solution:

The following changes were added to the parser `FunPar.fsl`

```
    %type <Absyn.expr list> AppExprList
    %type <string list> NAMELIST

    NAMELIST:
        NAME                {[$1]}
        | NAME NAMELIST         {$1::$2}
    ;
```

We have changed the AtExpr from

```
AtExpr:
    Const                               { $1                    }
  | NAME                                { Var $1                }
  | LET NAME EQ Expr IN Expr END        { Let($2, $4, $6)       }
  | LET NAME NAME EQ Expr IN Expr END   { Letfun($2, $3, $5, $7) }
  | LPAR Expr RPAR                      { $2                    }
;
```

to

```
AtExpr:
    Const                                 { $1                    }
  | NAME                                  { Var $1                }
  | LET NAME EQ Expr IN Expr END          { Let($2, $4, $6)       }
  | LET NAME NAMELIST EQ Expr IN Expr END { Letfun($2, $3, $5, $7) }
  | LPAR Expr RPAR                        { $2                    }
;
```

And finally added the `AppExprList` as well:

```
AppExprList:
    | AtExpr                    {[$1]}
    | AtExpr AppExprList                {$1::$2}
;
```