IMP

MODULE IMP-SYNTAX

```
SYNTAX AExp ::= Int
             | Id
             | AExp / AExp [strict( strict())]
             | AExp + AExp [strict( strict())]
             | (AExp) [bracket( bracket())]

SYNTAX BExp ::= Bool
             | AExp ≤ AExp [seqstrict( seqstrict())]
             | ! BExp [strict( strict())]
             | BExp && BExp [strict( strict(1))]
             | (BExp) [bracket( bracket())]

SYNTAX Block ::= {}
             | {Stmt}

SYNTAX Stmt ::= Block
             | Id = AExp ; [strict( strict(2))]
             | if (BExp)Block else Block [strict( strict(1))]
             | while (BExp)Block
             | Stmt Stmt

SYNTAX Pgm ::= int Ids ; Stmt

SYNTAX Ids ::= List{Id,","}
```

END MODULE

MODULE IMP

```
SYNTAX KResult ::= Int
                 | Bool
```

END MODULE