Recall the grammar

```
PROGRAM -> SEQU
SEQU    -> COMMAND | COMMAND ; SEQU
COMMAND -> id := EXPR |
   if EXPR then SEQU else SEQU endif |
   while EXPR do SEQU endwhile
EXPR    -> number | id
```

Give some examples of programs written in this language.

Draw parse trees for your programs.

Tree building routines

What kind of abstract syntax trees do we want?

What kind of tree building routines should be added to our grammar to build these?

```
PROGRAM -> SEQU

SEQU    -> COMMAND
    | COMMAND ; SEQU

COMMAND -> id := EXPR

    | if EXPR then SEQU else SEQU endif

    | while EXPR do SEQU endwhile

EXPR    -> number | id
```


Tree building routines

```
PROGRAM -> SEQU                        {
        PROGRAM.treeptr = SEQU.treeptr }

SEQU    -> COMMAND                     {
        SEQU.treeptr = COMMAND.treeptr }

  | COMMAND ; SEQU                     {
        SEQU.treeptr = makenode(sequ,
          COMMAND.treeptr, SEQU.treeptr); }

COMMAND -> id := EXPR {
        COMMAND.treeptr =
          makenode(':=',makeleaf(id), EXPR.treeptr); }

  | if EXPR then SEQU else SEQU endif   {
        COMMAND.treeptr = makenode(if,EXPR.treeptr,
          makenode(branch, SEQU1.treeptr,
            SEQU2.treeptr);}

  | while EXPR do SEQU endwhile         {
        COMMAND.treeptr =
           makenode(while,EXPR.treeptr,SEQU.treeptr); }

EXPR    -> number   {EXPR.treeptr = makeleaf(number)};

  | id              {EXPR.treeptr = makeleaf(id);}
```