Programming Language HW 2 Zewen Hua

△ <C>1 ::= do <C>2 while <be>

<C>2.labin := <C>1.labin + 2

<C>1.labout := <C>2.labout

<C>1.code :=

"L" <C>1.labin ":NOP"

<C>2.code

<be>.code

"BZ L" (<C>1.labin+1)

"BR L" <C>1.labin

"L" (<C>1.labin+1) ":NOP"

root code:

LOAD 0

STO i

L1: NOP

LOAD i

STO T1

LOAD 1

ADD T1

STO i

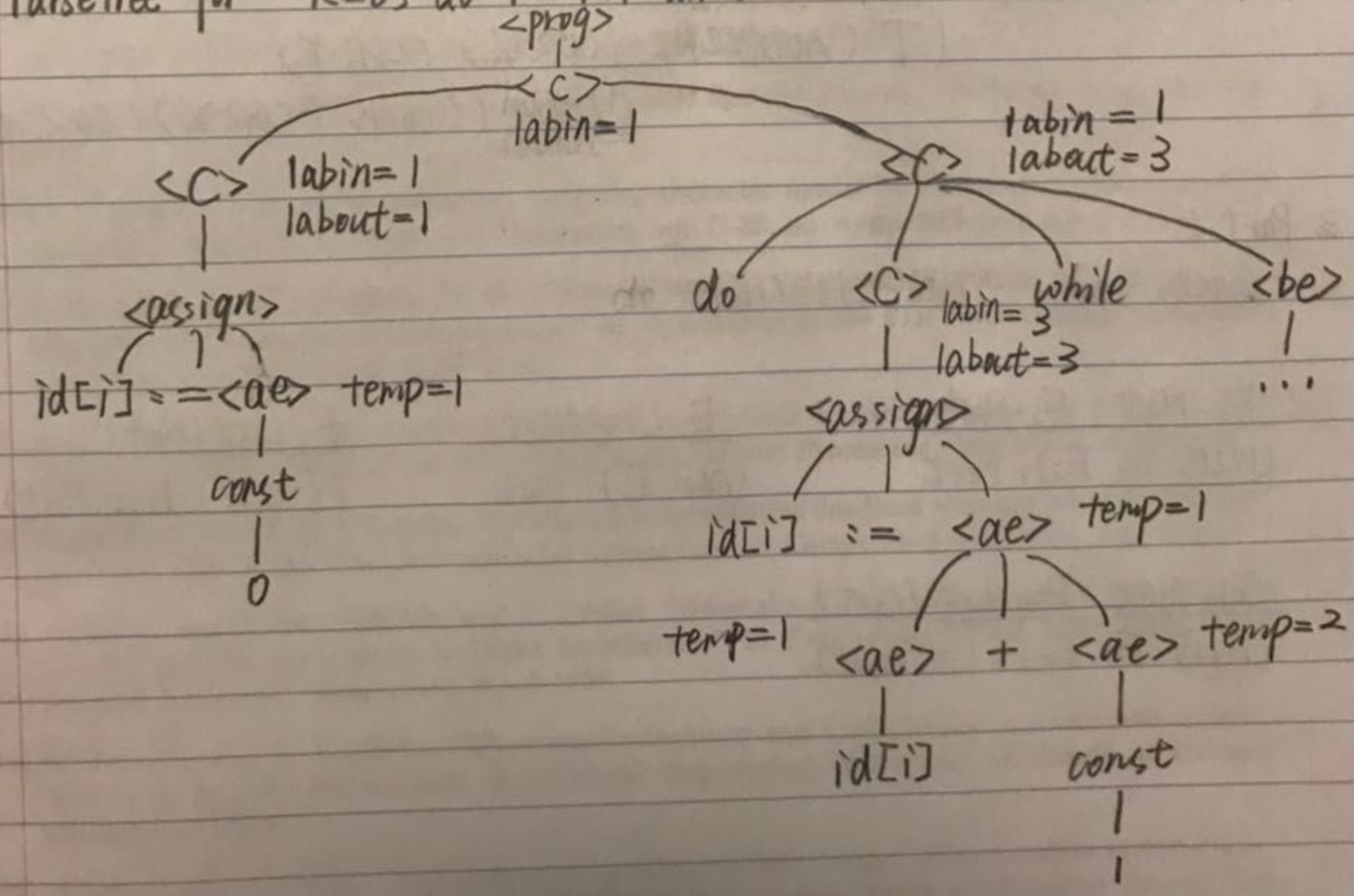
[…be…]

BZ L2

BR L1

L2 NOP.

ParseTree for i:=0; do i:=i+1 while …



2. bound(X, Z):

```
(DEFUN BOUND (X Z)
    (COND ((NULL Z) NIL)
          (T (COND ((EQ X (CAAR Z)) T)
          (T (BOUND X (CDR Z)))))))
```

getval(X, Z):

```
(DEFUN GETVAL (X Z)
    (COND ((NULL Z) NIL)
          (T (COND ((EQ X (CAAR Z)) (CDAR Z))
          (T (GETVAL X (CDR Z)))))))
```

addpairs(X, Y, Z):

```
(DEFUN ADDPAIRS (X Y Z)
    (COND ((NULL X) Z)
          (T (ADDPAIRS (CDR X) (CDR Y)
                       (CONS ((CONS (CAR X) (CAR Y)) Z)))))
```

3. Part 1

const: Nat NIL: List(Nat)

$$\frac{E_1: Nat \quad E_2: Nat}{(PLUS\ E_1\ E_2): Nat} \qquad \frac{E: List(Nat)}{(CAR\ E): Nat} \qquad \frac{E: List(Nat)}{(CDR\ E): List(Nat)}$$

$$\frac{E_1: Nat \quad E_2: List(Nat)}{(CONS\ E_1\ E_2): List(Nat)}$$

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{2: Nat \quad 3: Nat}{(PLUS\ 2\ 3): Nat} \quad NIL: List(Nat)}{(CONS\ (PLUS\ 2\ 3)\ NIL): List(Nat)} \quad 1: Nat}{(CONS\ 1\ (CONS\ (PLUS\ 2\ 3)\ NIL)): List(Nat)}}{(CDR\ (CONS\ 1\ (CONS\ (PLUS\ 2\ 3)\ NIL))): List(Nat)}}{(CAR\ (CDR\ (CONS\ 1\ (CONS\ (PLUS\ 2\ 3)\ NIL)))): Nat}$$

3. Part 2

Attribute: <E>.type

Evaluation rules:

<E>::= const
<E>.type = Nat
| NIL
<E>.type = List(Nat)
| (PLUS <E1> <E2>
<E>.type = Nat
cond: <E1>.type = Nat and <E2>.type = Nat
| (CAR <E1>)
<E>.type = Nat
cond: <E1>.type = List(Nat)
| (CDR <E1>)
<E>.type = List(Nat)
cond: <E1>.type = List(Nat)
| (CONS <E1> <E2>)
<E>.type = List(Nat)
cond: <E1>.type = Nat and <E2>.type = Nat

4. <E>::= (RAND <E1> <E2>)
<E>.type = <E1>.type
cond: <E1>.type = <E2>.type
<E1>.type = Nat or <E1>.type = List(Nat)

$$\frac{E_1 : T \quad E_2 : T}{(\text{RAND } E_1\ E_2) : T} \qquad T : \text{Nat or List(Nat)}$$