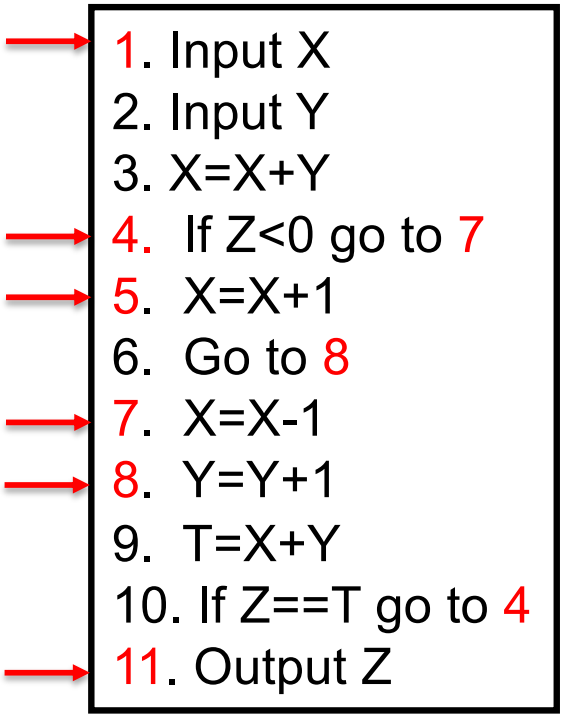
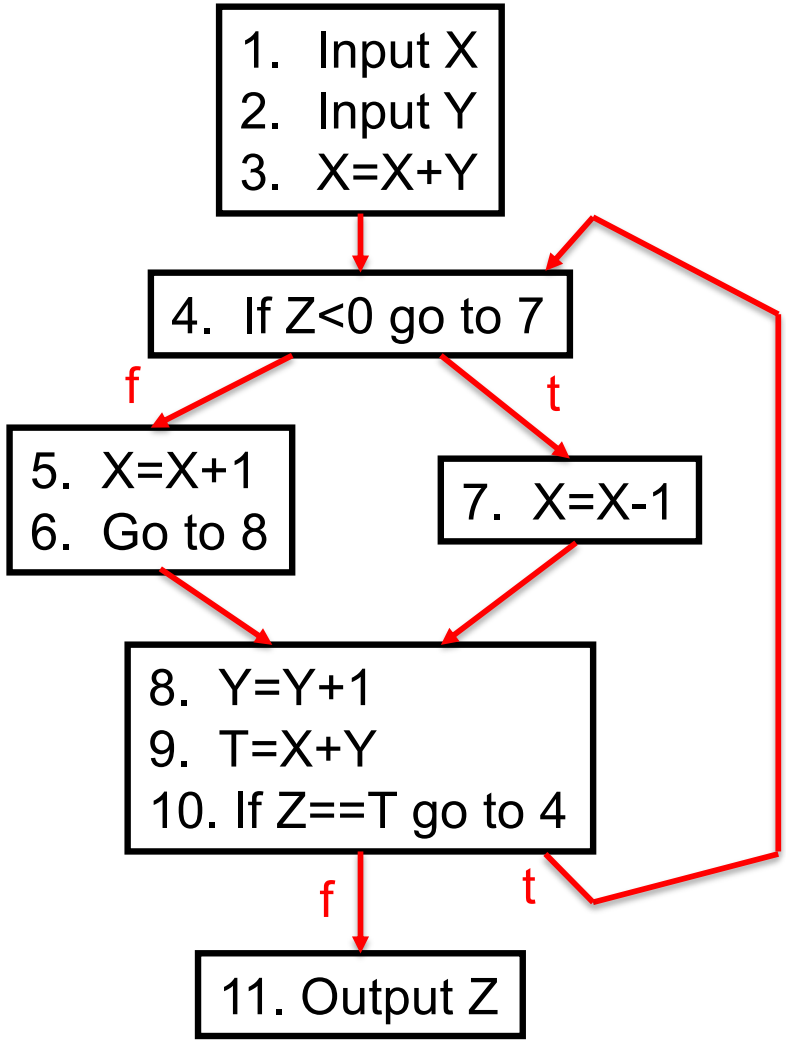
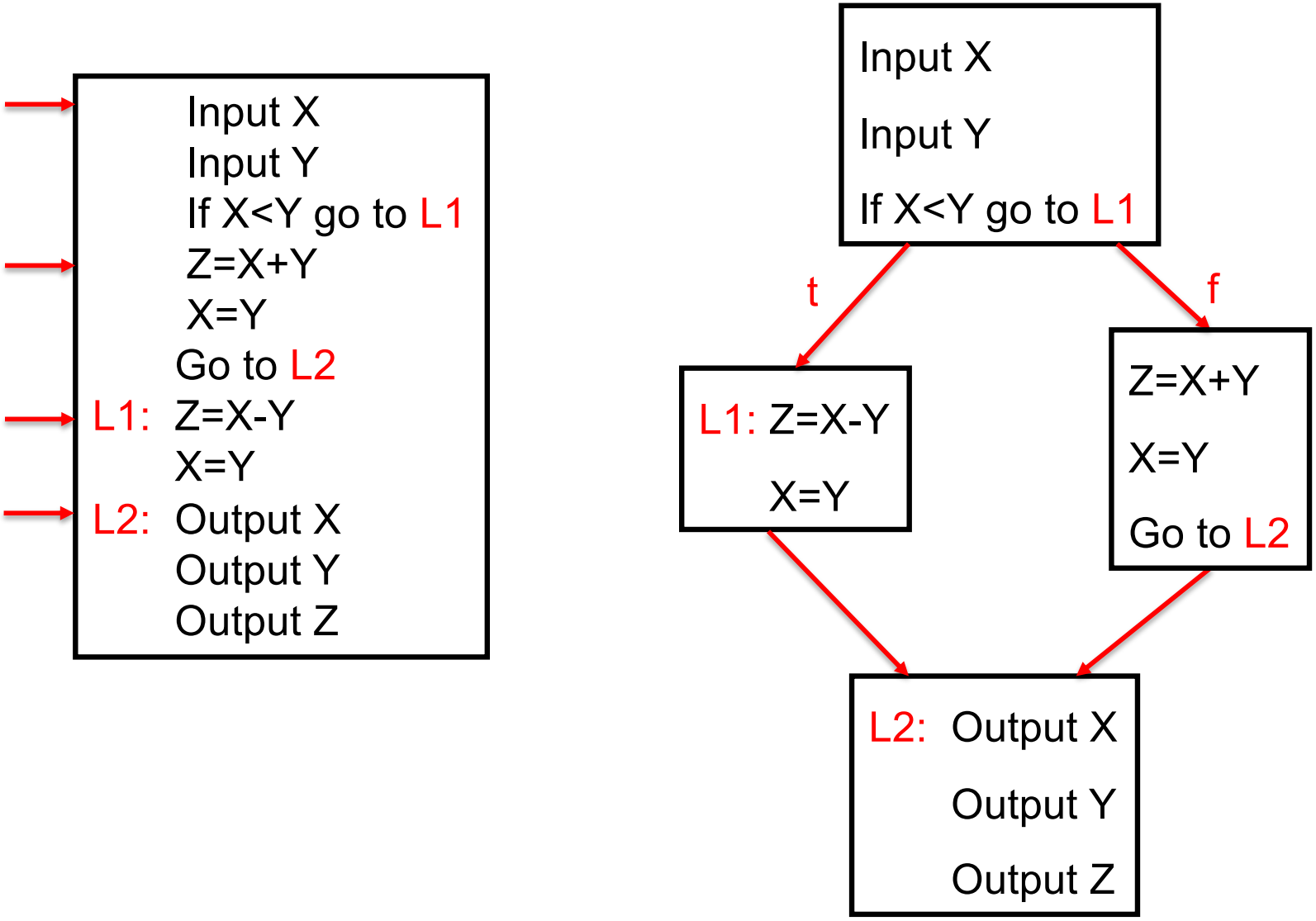
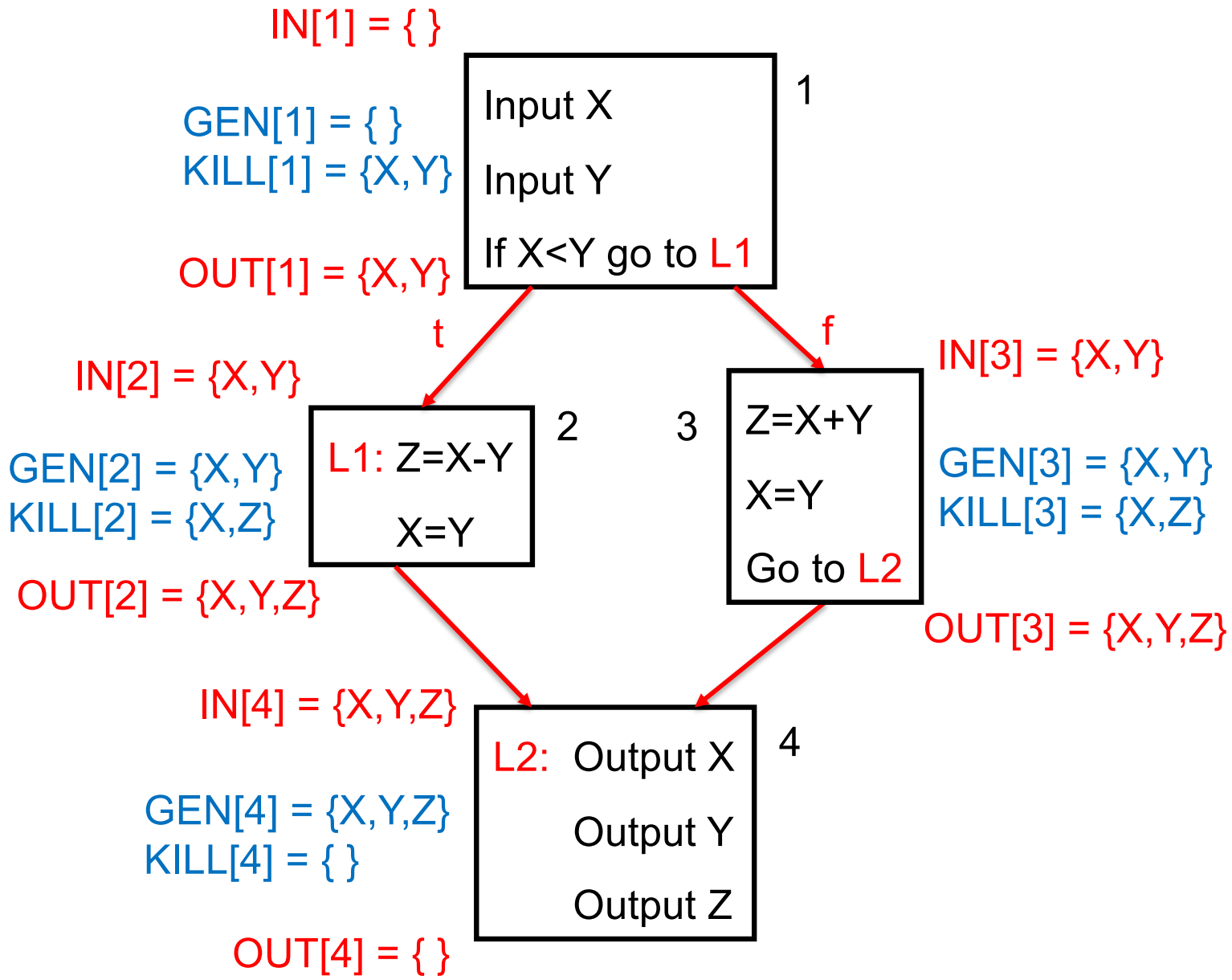
Sample Problems for Review

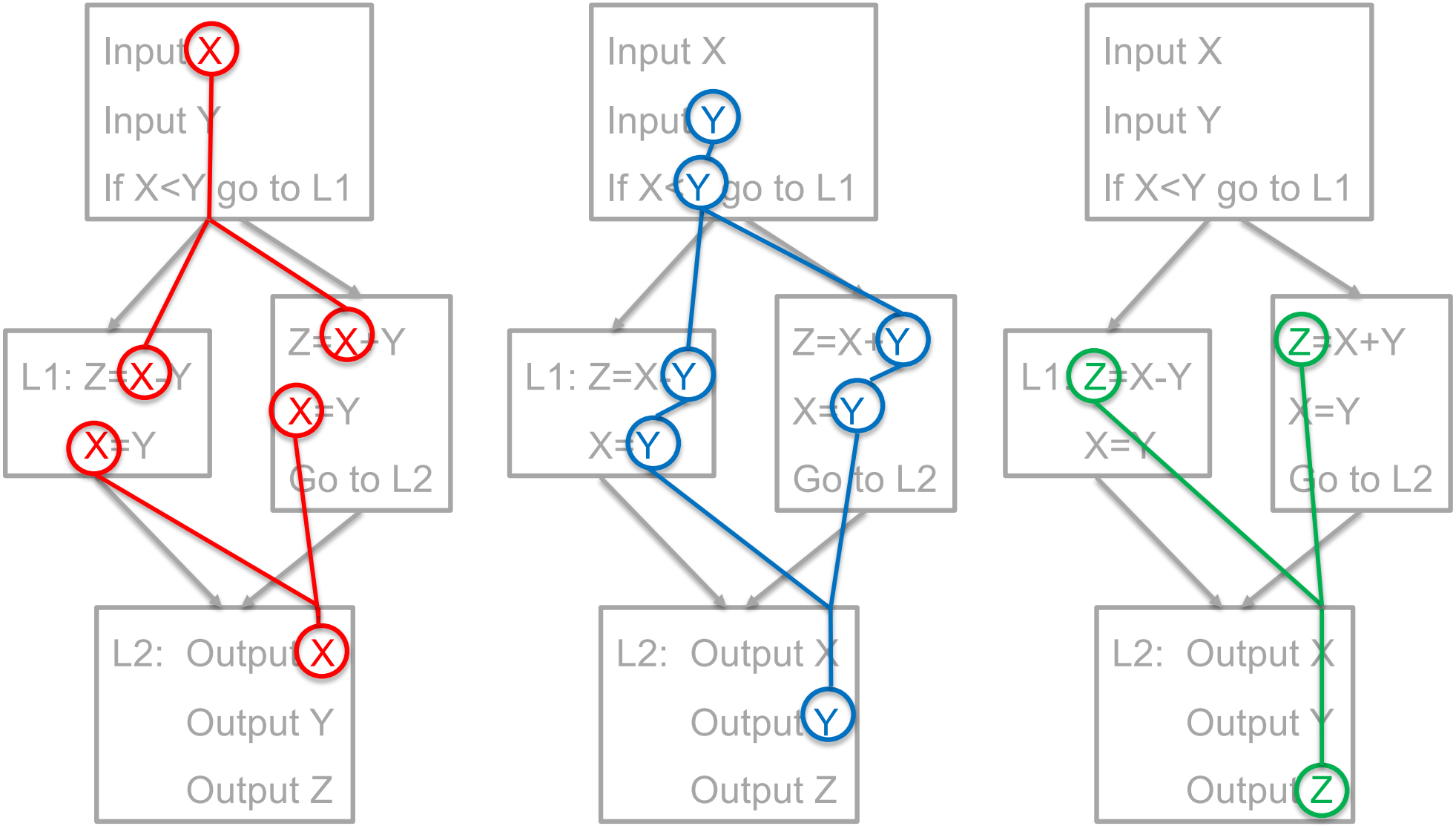

1. Input X
2. Input Y
3. X=X+Y
4. If Z<0 go to 7
5. X=X+1
6. Go to 8
7. X=X-1
8. Y=Y+1
9. T=X+Y
10. If Z==T go to 4
11. Output Z


1. Input X
2. Input Y
3. X=X+Y
4. If Z<0 go to 7
f
t
5. X=X+1
6. Go to 8
7. X=X-1
8. Y=Y+1
9. T=X+Y
10. If Z==T go to 4
f
t
11. Output Z

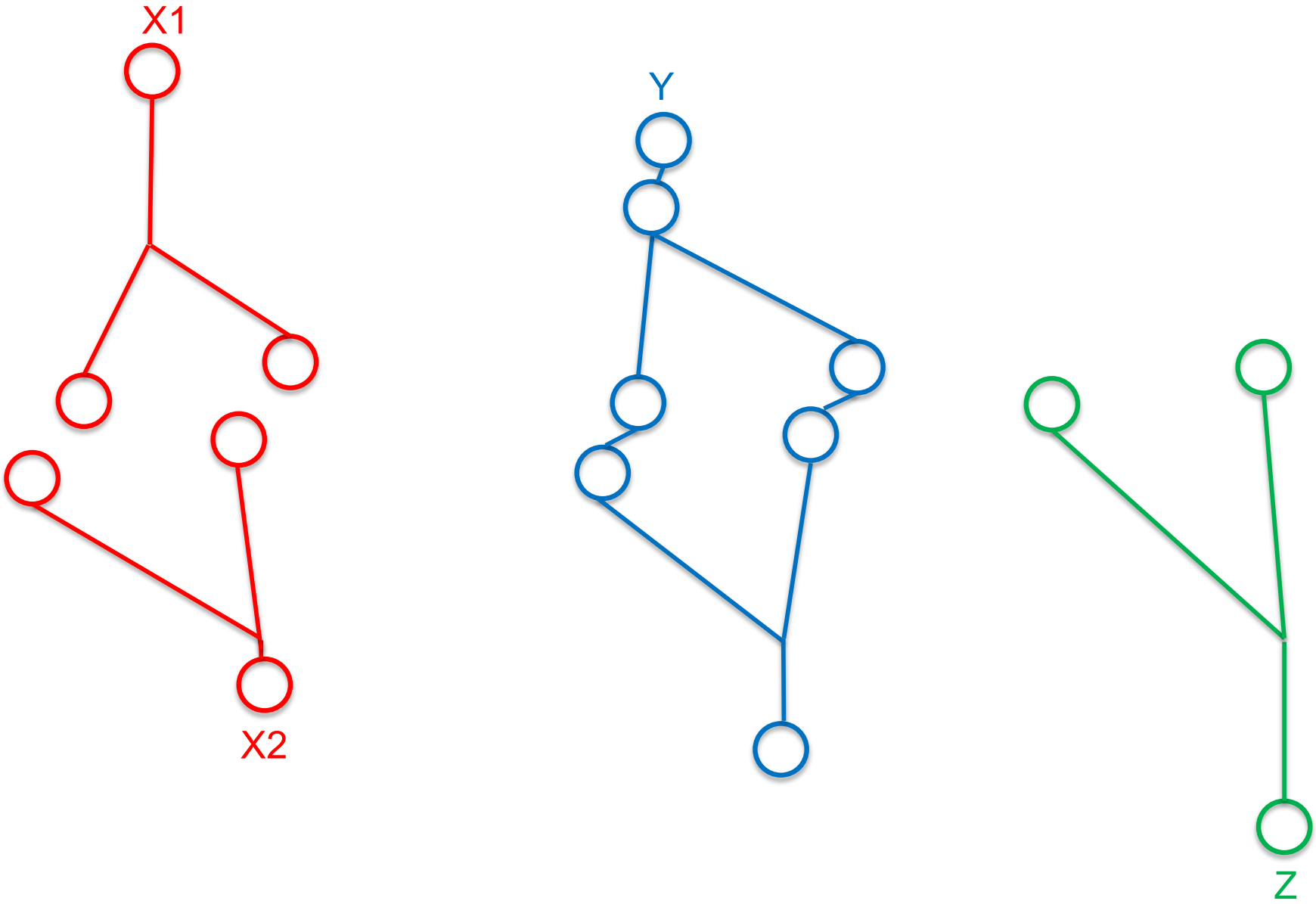

Input X
Input Y
If X<Y go to L1
Z=X+Y
X=Y
Go to L2
L1: Z=X-Y
X=Y
L2: Output X
Output Y
Output Z
Input X
Input Y
If X<Y go to L1
t
f
L1: Z=X-Y
X=Y
Z=X+Y
X=Y
Go to L2
L2: Output X
Output Y
Output Z


IN[1] = { }
Input X
Input Y
If X<Y go to L1
1
GEN[1] = { }
KILL[1] = {X,Y}
OUT[1] = {X,Y}
t
f
IN[2] = {X,Y}
L1: Z=X-Y
X=Y
2
3
Z=X+Y
X=Y
Go to L2
IN[3] = {X,Y}
GEN[2] = {X,Y}
KILL[2] = {X,Z}
OUT[2] = {X,Y,Z}
GEN[3] = {X,Y}
KILL[3] = {X,Z}
OUT[3] = {X,Y,Z}
IN[4] = {X,Y,Z}
L2: Output X
Output Y
Output Z
4
GEN[4] = {X,Y,Z}
KILL[4] = { }
OUT[4] = { }

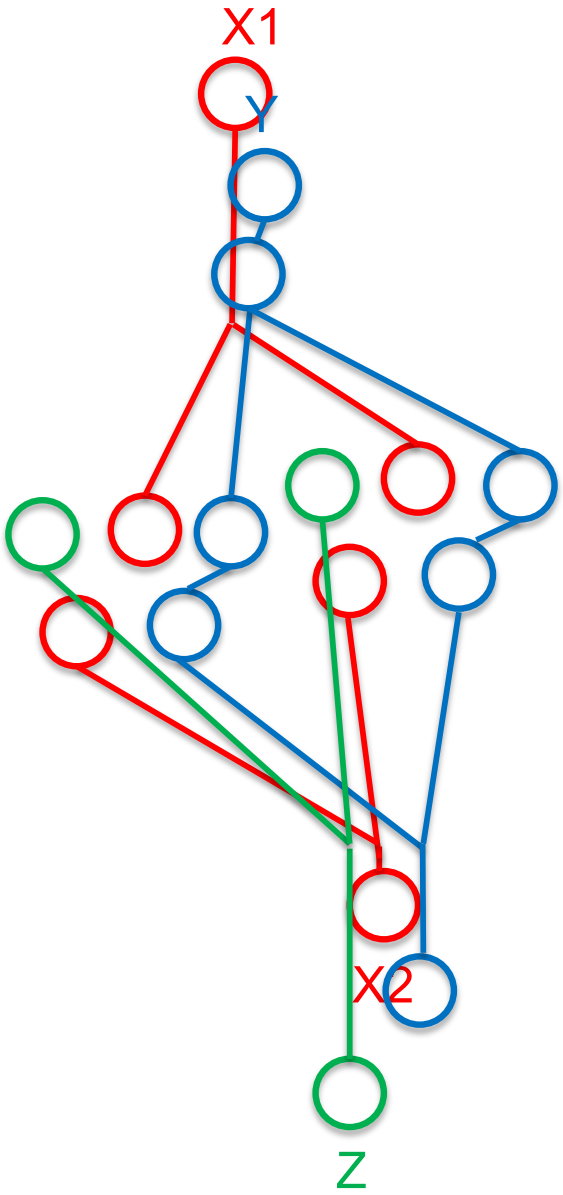
LIVE RANGES OF X, Y and Z

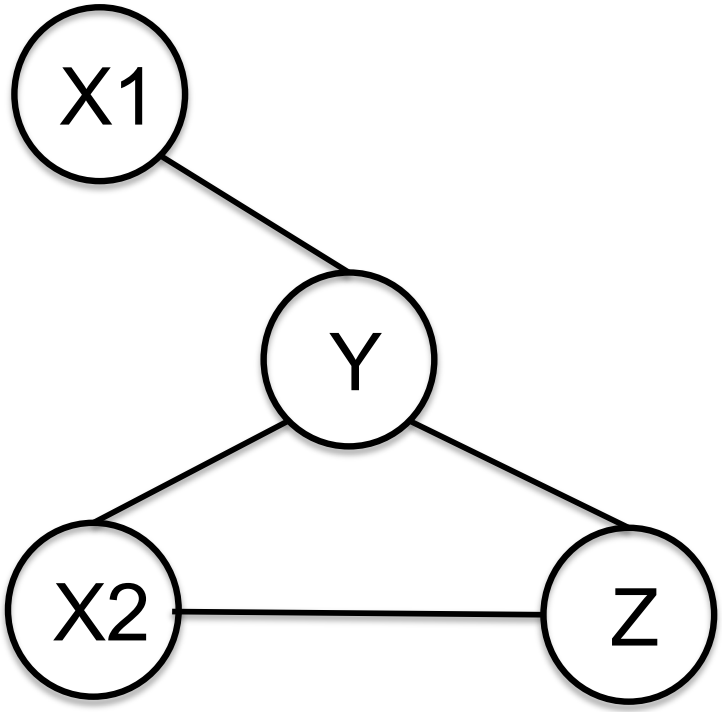


LIVE RANGES OF X, Y and Z

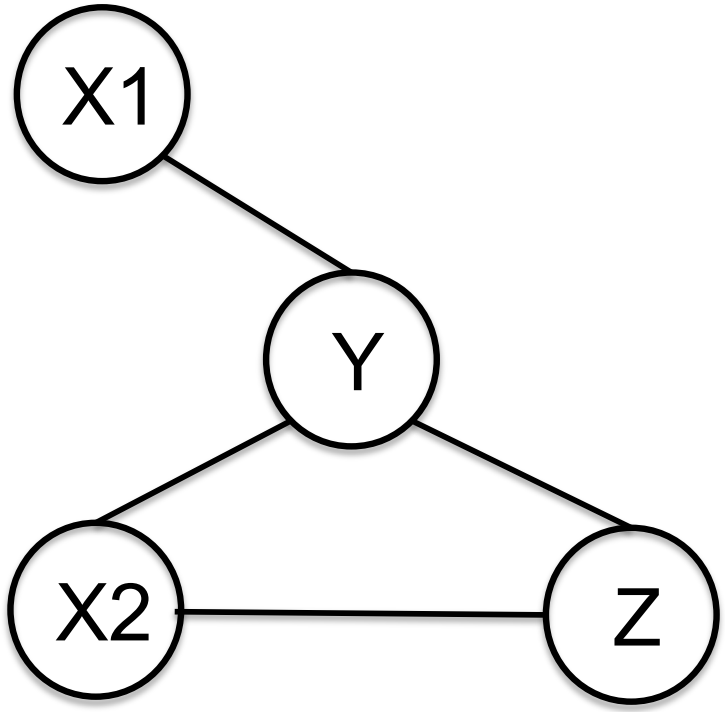


INTERFERENCE GRAPH





REGISTER ALLOCATION: R1, R2, R3



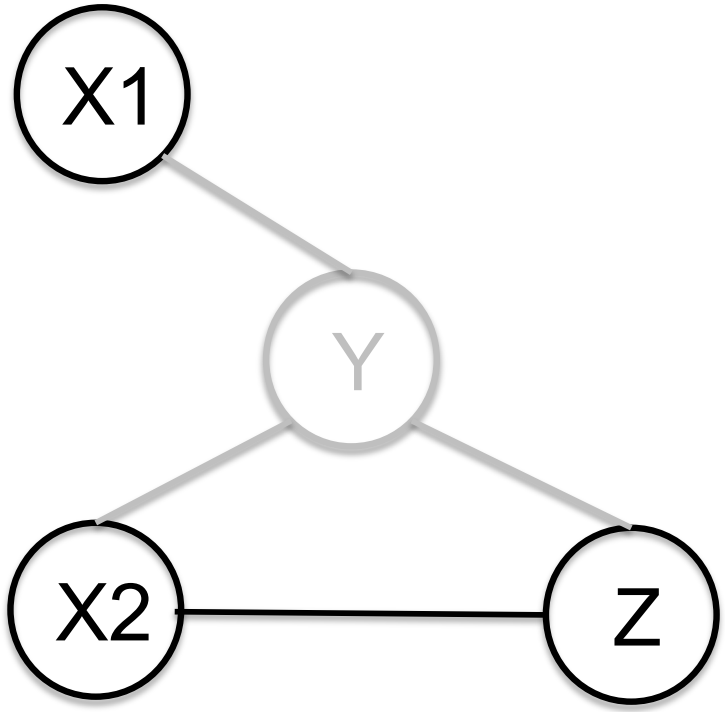
REMOVE DEGREE<3

X1, X2, Z; Y

COLOR IN REVERSE ORDER

Y	R1
Z	R2
X2	R3
X1	R2 or R3

REGISTER ALLOCATION: R1, R2



REMOVE DEGREE<2

X1; spill Y; X2, Z

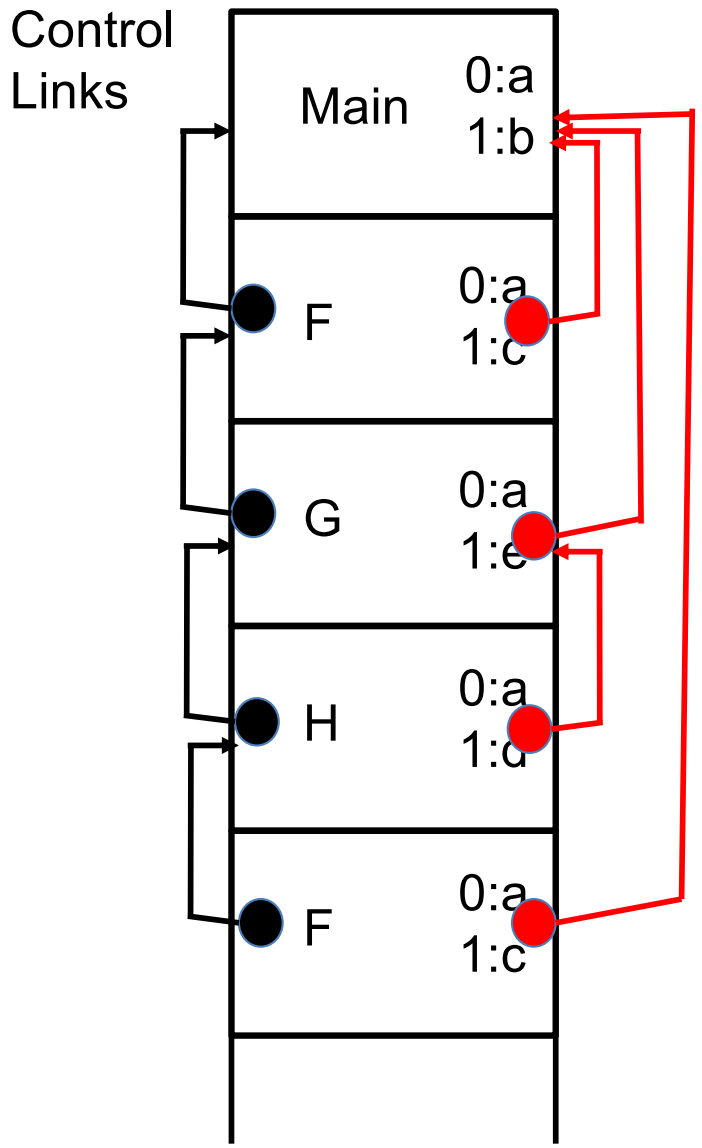
COLOR IN REVERSE ORDER

Z	R1
X2	R2
X1	R1 or R2

```
0  Main () {
        Int a, b;

   1  F() {
           Int a, c;
        2 Call G();
      }

   1 G() {
           Int a, e;
        2  H() {
              Int a, d;

           3 Call F();
           }

        2 Call H();
      }

   1 Call F()
}
```

Main → F → G → H → F

c-d → 1-1 2-1 2-2 3-1



Access
b in H
traverse
3-1=2 links
then at
offset at of 1
find b

CONSTRUCT

```
if x < y then
      <otherstatements>
elseif a > b then
      <otherstatements>
……
elseif c == d then
      <otherstatements>
else
      <otherstatements>
endif
```

GRAMMAR
RELEVANT PRODUCTIONS

<S> → if <condt> then <otherstatements> <rest>

<rest> → elseif <condt> then <otherstatements> <rest>
| else <otherstatements> endif

<condt> → id relop id

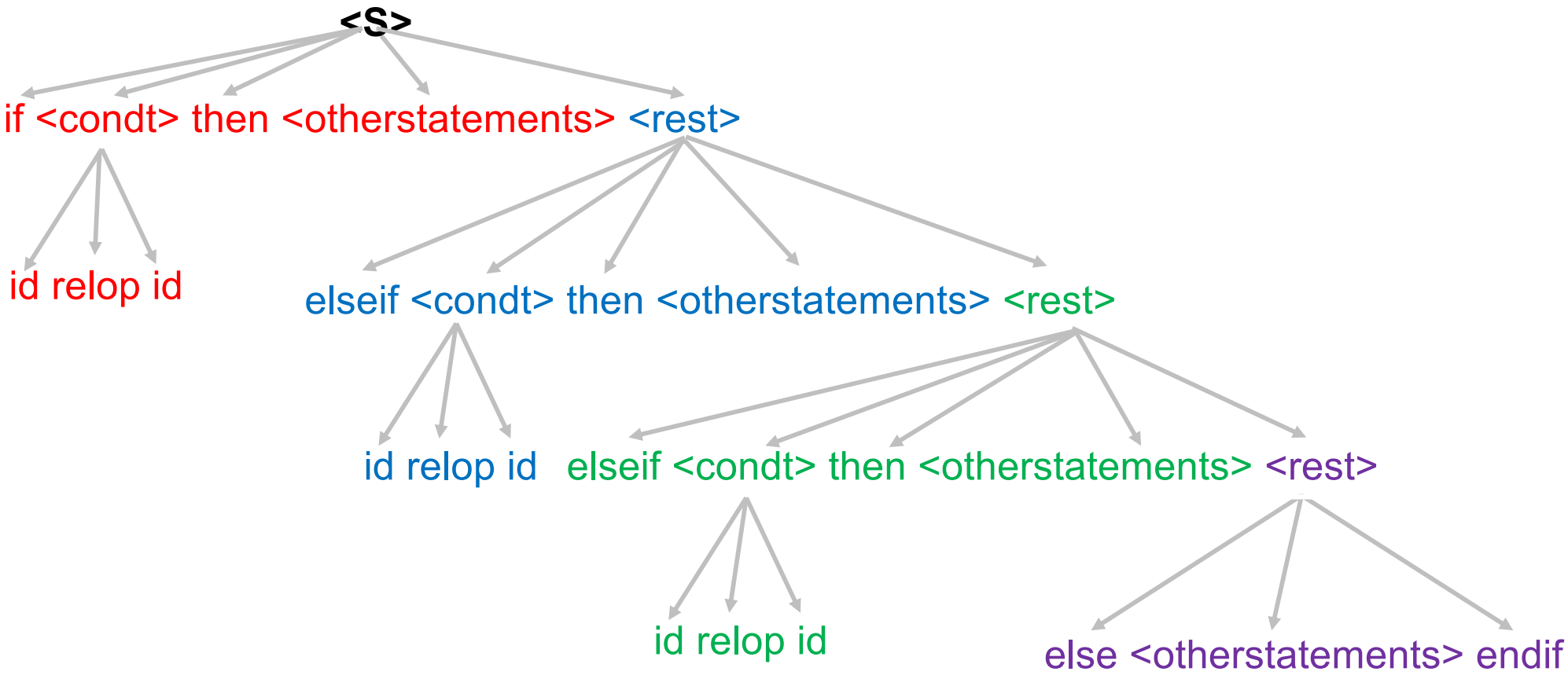
Question:
Provide SEMANTIC RULES that generate code and finally place it in attribute **<S>.code**

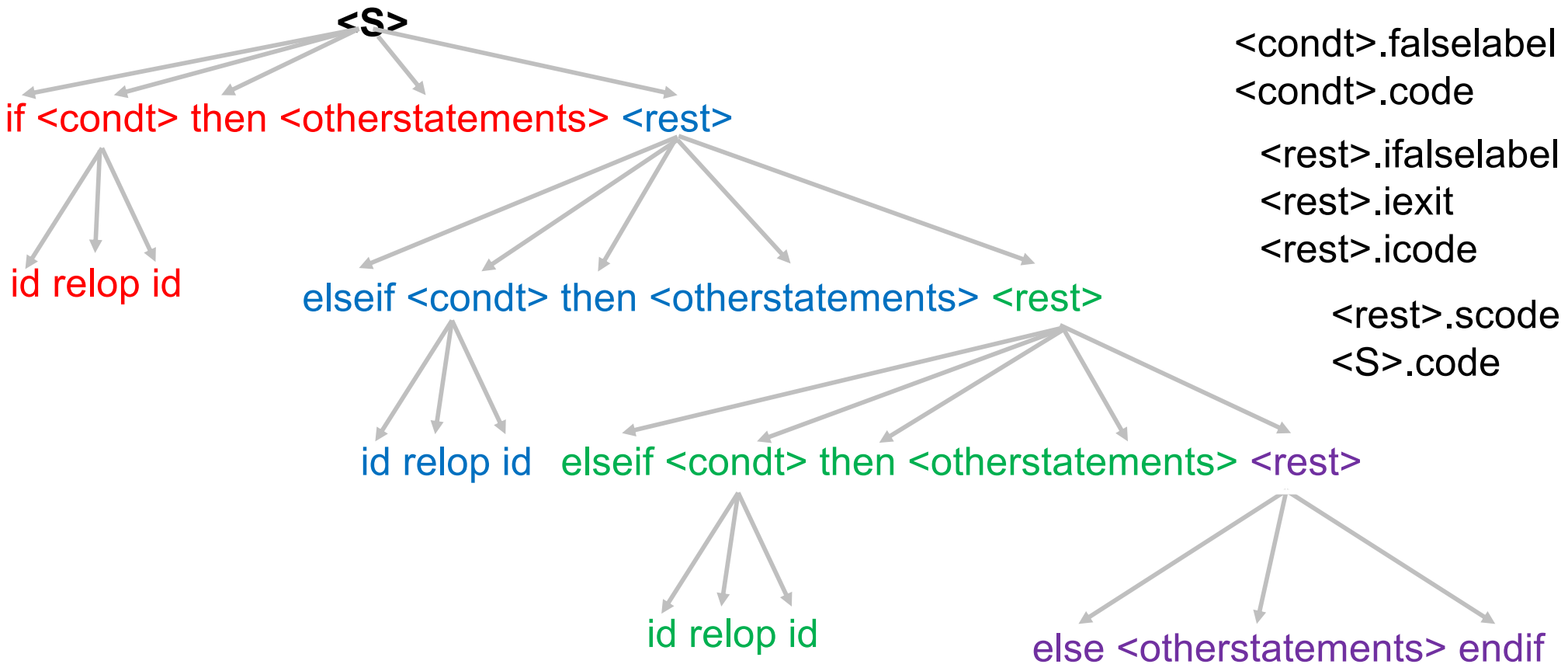
CONSTRUCT

```
if x < y then
        <otherstatements>
elseif a > b then
        <otherstatements>
elseif c == d then
        <otherstatements>
else
        <otherstatements>
endif
……
```

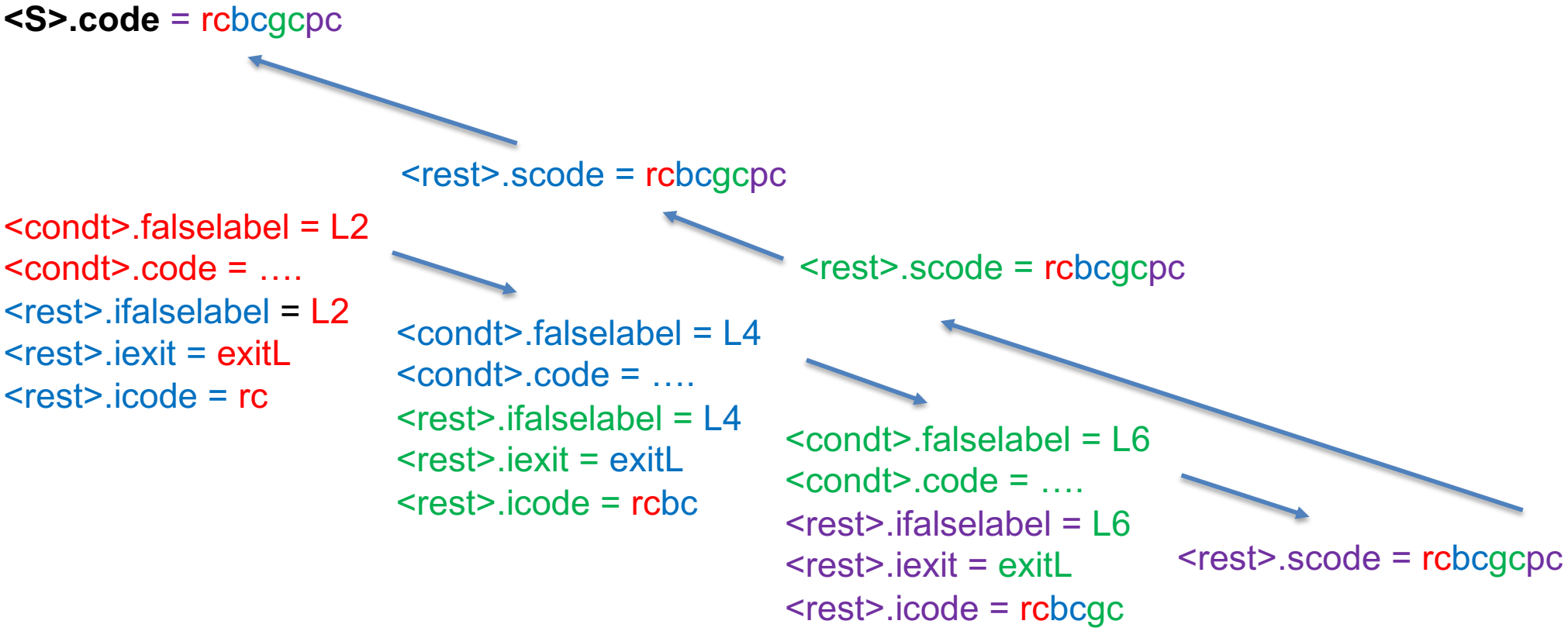
INTERMEDIATE CODE

```
        if x < y go to L1
        go to L2
L1: <otherstatements>
        go to exitL
L2: If a > b go to L3
        go to L4
L3: <otherstatements>
        go to exitL
L4: if c==d go to L5
        go to L6
L5: <otherstatements>
        go to exitL
L6: <otherstatements>
exitL: …….
```


<S>
if <condt> then <otherstatements> <rest>
id relop id
elseif <condt> then <otherstatements> <rest>
id relop id
elseif <condt> then <otherstatements> <rest>
id relop id
else <otherstatements> endif



if x < y go to L1 go to L2 L1: <otherstatements> go to exitL L2: <rest>	L2: If a > b go to L3 go to L4 L3: <otherstatements> go to exitL L4: <rest>	L4: if c==d go to L5 go to L6 L5: <otherstatements> go to exitL L6: <rest>	L6: <otherstatements> exitL: …….



if x < y go to L1	L2: If a > b go to L3	L4: if c==d go to L5	L6: <otherstatements>
go to L2	go to L4	go to L6	exitL: …….
L1: <otherstatements>	L3: <otherstatements>	L5: <otherstatements>	
go to exitL	go to exitL	go to exitL	
L2: <rest>	L4: <rest>	L6: <rest>	

<condt> → id_1 relop id_2 {
truelabel = newlabel();
<condt>.falselabel = newlabel();
<condt>.code = gen("if" id_1.place "relop" id_2.place "go to" truelabel)
|| gen("go to" <condt>.falselabel) || gen(truelabel":")
}

<S> → if <condt> then <otherstatements>
{
<rest>.ifalselabel = <condt>.falselabel;
<rest>.iexit = newlabel();
<rest>.icode = <condt>.code || <otherstatements>.code ||
gen("go to" <rest>.iexit)
}
<rest> { <S>.code = <rest>.scode }

<rest$_1$> → elseif <condt> then <otherstatements>

{

<rest$_2$>.icode = <rest$_1$>.icode || gen(<rest$_1$>.ifalselabel ":") ||

<condt>.code || <otherstatements>.code || gen("go to" <rest$_1$>.iexit);

<rest$_2$>.ifalselabel = <condt>.falselabel;

<rest$_2$>.iexit = <rest$_1$>.iexit

}

<rest$_2$> { <rest$_1$>.scode = <rest$_2$>.scode }

<rest$_1$> → else <otherstatements> endif

{

<rest$_1$>.scode = <rest$_1$>.icode || gen(<rest$_1$>.ifalselabel ":")

|| <otherstatements>.code || gen(<rest$_1$>.iexit ":")

}