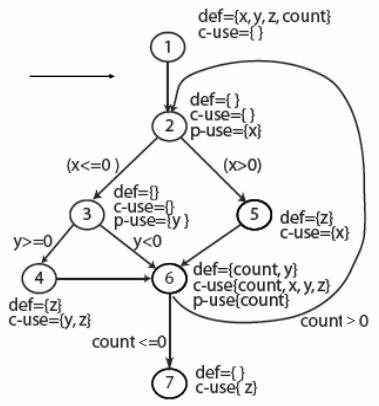
Exercise 1

Def-clear path (another example)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

Q1: Find def-clear paths for defs and uses of *x* and *z*.
Q2: Which definitions are live at node 4?

Def-clear Path

Variable x:

$d_1(x)$ and $u_2(x)$:	1->2
$d_1(x)$ and $u_5(x)$:	1->2->5 or 1->2->5->6->2->5
$d_1(x)$ and $u_6(x)$:	1->2->3->6
	1->2->3->4->6
	1->2->5>6

Variable z:

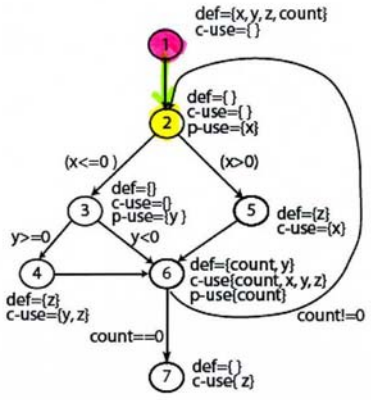
$d_1(z)$ and $u_4(z)$:	1->2->3->4
$d_1(z)$ and $u_6(z)$:	1->2->3->6
$d_4(z)$ and $u_6(z)$:	4->6
$d_5(z)$ and $u_6(z)$:	5->6
$d_4(z)$ and $u_4(z)$:	4->6->2->3->4
$d_5(z)$ and $u_4(z)$:	5->6->2->3->4
$d_4(z)$ and $u_7(z)$:	4->6->7 or 4->6->2->3->6->7
$d_5(z)$ and $u_7(z)$:	5->6->7
$d_1(z)$ and $u_7(z)$:	1->2->3->6->7

If loops are considered, there can be more def-clear paths.

3

$d_1(x), u_2(x)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



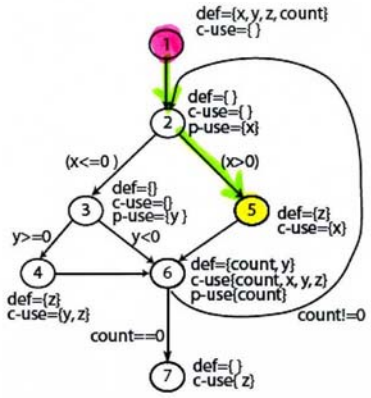
Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

4

du(x, us(x)

(does not go through the loop

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x<=0) {
7         if (y>=0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



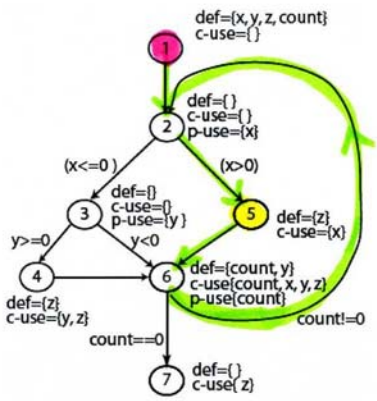
Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

5

du(x, us(x)

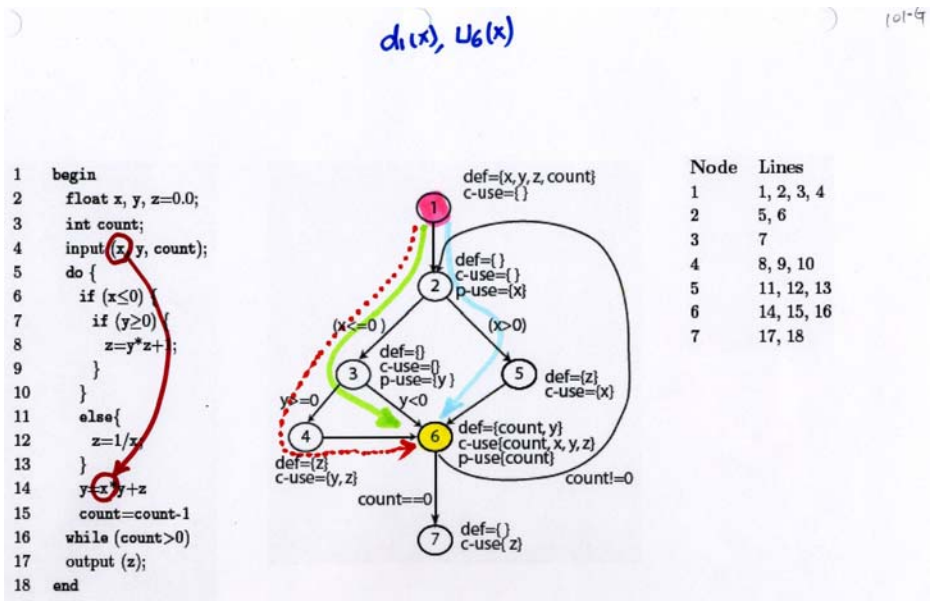
(going through the loop once)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x<=0) {
7         if (y>=0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

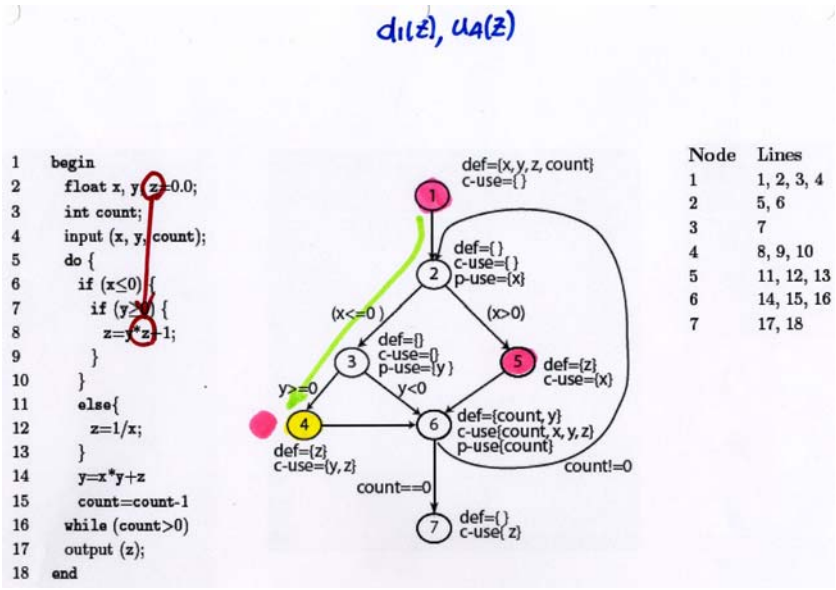


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

6


$d_1(x)$, $u_6(x)$
1 begin
2 float x, y, z=0.0;
3 int count;
4 input (x, y, count);
5 do {
6 if (x≤0) {
7 if (y≥0) {
8 z=y*z+1;
9 }
10 }
11 else{
12 z=1/x;
13 }
14 y=x*y+z
15 count=count-1
16 while (count>0)
17 output (z);
18 end
def={x, y, z, count}
c-use={ }
def={ }
c-use={ }
p-use={x}
(x<=0)
(x>0)
def={ }
c-use={ }
p-use={y }
def={z}
c-use={x}
y>=0
y<0
def={count, y}
c-use{count, x, y, z}
p-use{count}
def={z}
c-use={y, z}
count!=0
count==0
def={ }
c-use{ z}
Node Lines
1 1, 2, 3, 4
2 5, 6
3 7
4 8, 9, 10
5 11, 12, 13
6 14, 15, 16
7 17, 18

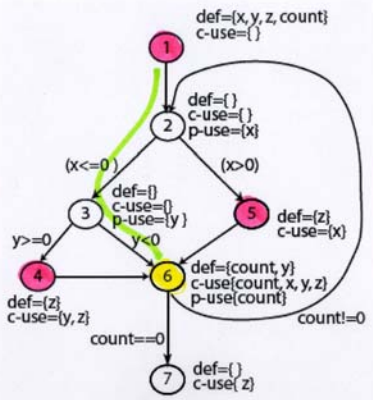
7


$d_1(z)$, $u_4(z)$
1 begin
2 float x, y, z=0.0;
3 int count;
4 input (x, y, count);
5 do {
6 if (x≤0) {
7 if (y≥0) {
8 z=y*z+1;
9 }
10 }
11 else{
12 z=1/x;
13 }
14 y=x*y+z
15 count=count-1
16 while (count>0)
17 output (z);
18 end
def={x, y, z, count}
c-use={ }
def={ }
c-use={ }
p-use={x}
(x<=0)
(x>0)
def={ }
c-use={ }
p-use={y }
def={z}
c-use={x}
y>=0
y<0
def={count, y}
c-use{count, x, y, z}
p-use{count}
def={z}
c-use={y, z}
count!=0
count==0
def={ }
c-use{ z}
Node Lines
1 1, 2, 3, 4
2 5, 6
3 7
4 8, 9, 10
5 11, 12, 13
6 14, 15, 16
7 17, 18

8

$d_1(z)$, $U_6(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z;
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

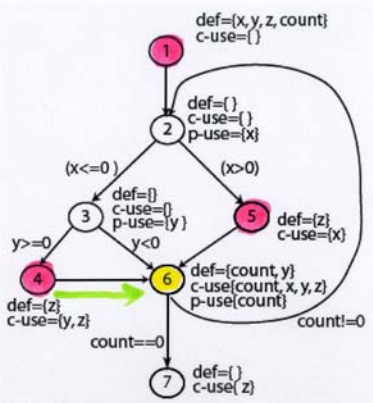


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

9

$d_4(z)$, $U_6(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z;
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

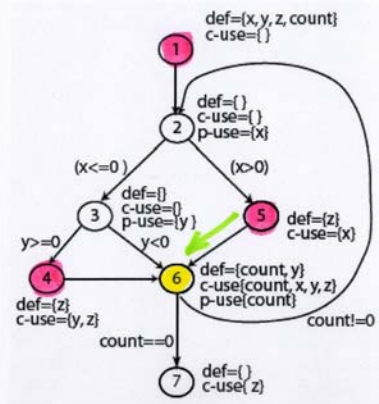


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

10

$d_6(z), u_6(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z;
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

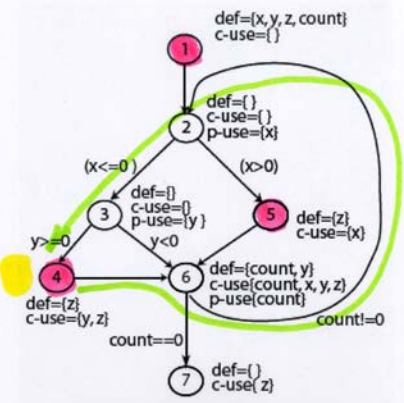


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

11

$d_4(z), u_4(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

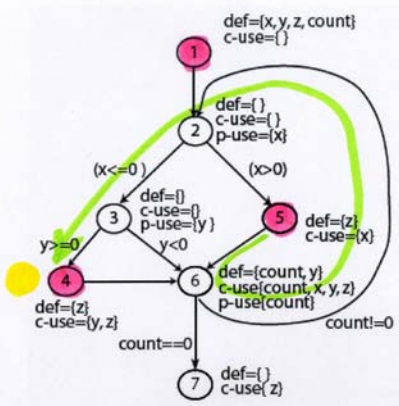


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

12

$d_5(z)$, $u_4(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



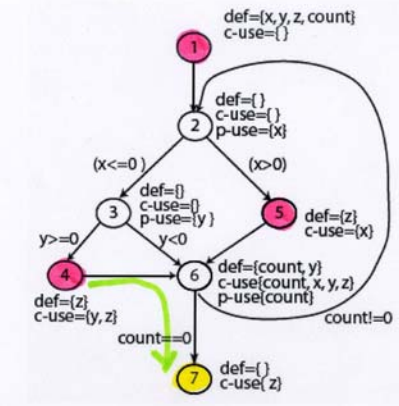
Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

13

$d_4(z)$, $u_7(z)$

(does not go through the loop)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

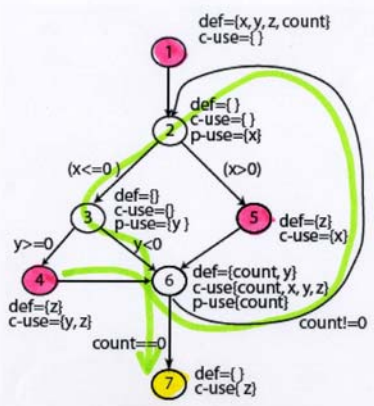
14

101-O

$d_4(z), U_\eta(z)$

going through the loop once

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



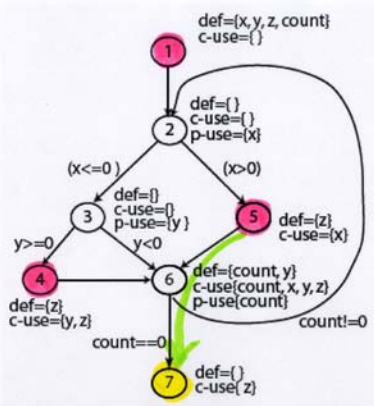
Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

15

101-P

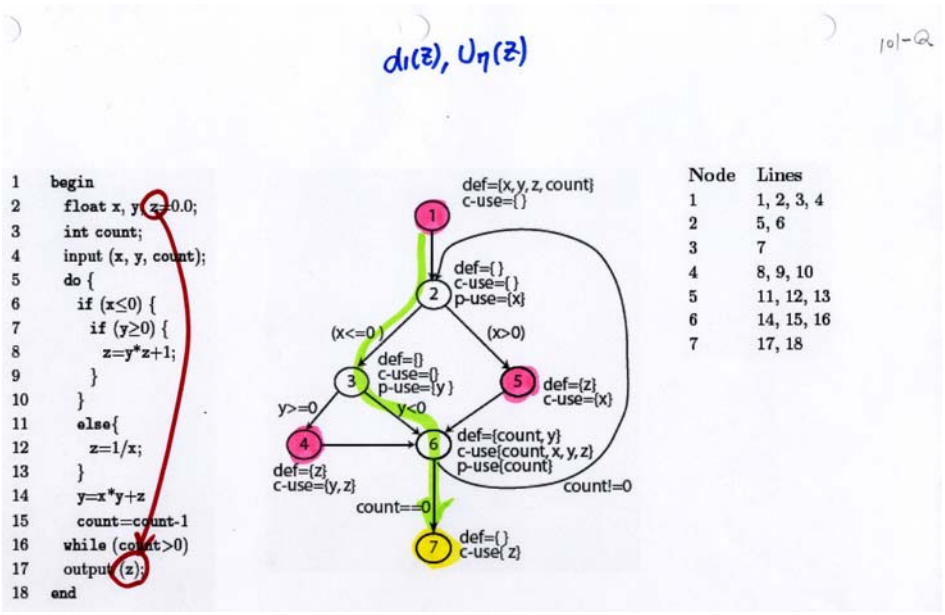
$d_5(z), U_\eta(z)$

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

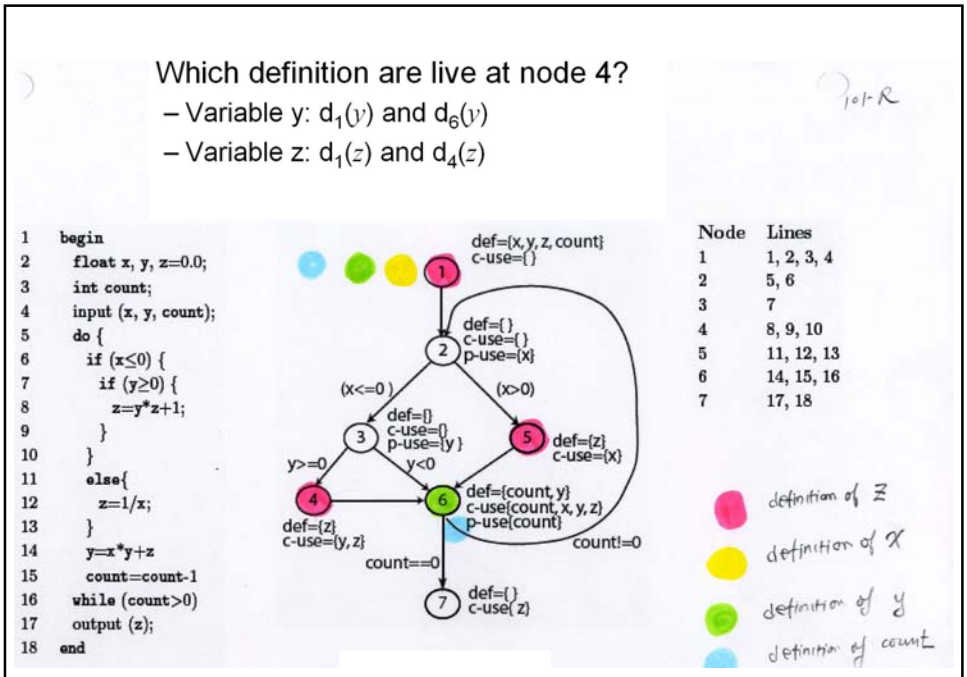


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

16


$d_1(z), U_\eta(z)$
101-Q
1 begin
2 float x, y, z=0.0;
3 int count;
4 input (x, y, count);
5 do {
6 if (x≤0) {
7 if (y≥0) {
8 z=y*z+1;
9 }
10 }
11 else{
12 z=1/x;
13 }
14 y=x*y+z
15 count=count-1
16 while (count>0)
17 output (z);
18 end
def={x, y, z, count}
c-use={ }
def={ }
c-use={ }
p-use={x}
(x<=0)
(x>0)
def={}
c-use={}
p-use={y }
def={z}
c-use={x}
y>=0
y<0
def={count, y}
c-use{count, x, y, z}
p-use{count}
def={z}
c-use={y, z}
count!=0
count==0
def={ }
c-use{ z}
Node Lines
1 1, 2, 3, 4
2 5, 6
3 7
4 8, 9, 10
5 11, 12, 13
6 14, 15, 16
7 17, 18

17

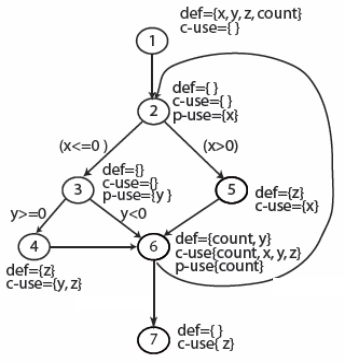

Which definition are live at node 4?
– Variable y: $d_1(y)$ and $d_6(y)$
– Variable z: $d_1(z)$ and $d_4(z)$
101-R
1 begin
2 float x, y, z=0.0;
3 int count;
4 input (x, y, count);
5 do {
6 if (x≤0) {
7 if (y≥0) {
8 z=y*z+1;
9 }
10 }
11 else{
12 z=1/x;
13 }
14 y=x*y+z
15 count=count-1
16 while (count>0)
17 output (z);
18 end
def={x, y, z, count}
c-use={ }
def={ }
c-use={ }
p-use={x}
(x<=0)
(x>0)
def={}
c-use={}
p-use={y }
def={z}
c-use={x}
y>=0
y<0
def={count, y}
c-use{count, x, y, z}
p-use{count}
def={z}
c-use={y, z}
count!=0
count==0
def={ }
c-use{ z}
Node Lines
1 1, 2, 3, 4
2 5, 6
3 7
4 8, 9, 10
5 11, 12, 13
6 14, 15, 16
7 17, 18
definition of z
definition of x
definition of y
definition of count

18

Exercise 2

19

Def-use pairs (example)



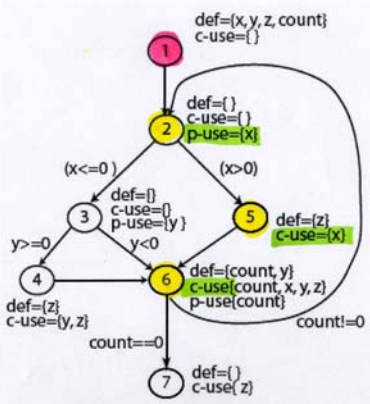
Variable (v)	Defined in node (n)	dcu (v, n)	dpu (v, n)
x	1	{5, 6}	{(2, 3), (2, 5)}
y	1	{4, 6}	{(3, 4), (3, 6)}
y	6	{4, 6}	{(3, 4), (3, 6)}
z	1	{4, 6, 7}	{ }
z	4	{4, 6, 7 }	{ }
z	5	{4, 6, 7}	{ }
count	1	{6}	{(6, 2), (6, 7) }
count	6	{6}	{(6, 2), (6, 7) }

Variable z: (5, 4)
Variable count: (1, (6,2)), and (1, (6,7))
are infeasible.

20

With resp to x

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

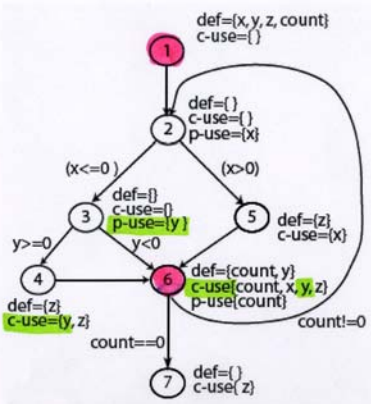


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

21

w.r.t the definition of y at (1)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

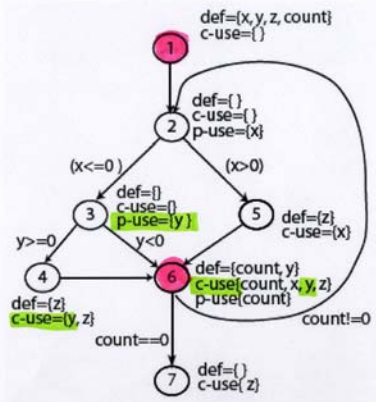


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

22

w.r.t the definition of y at (6)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

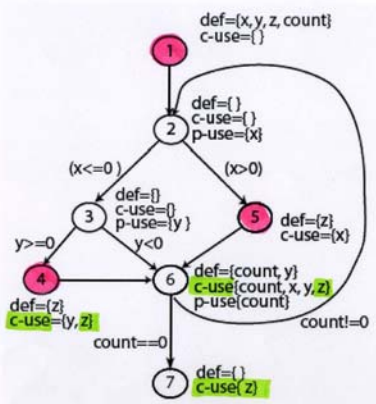


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

23

w.r.t the definition of z at (1)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

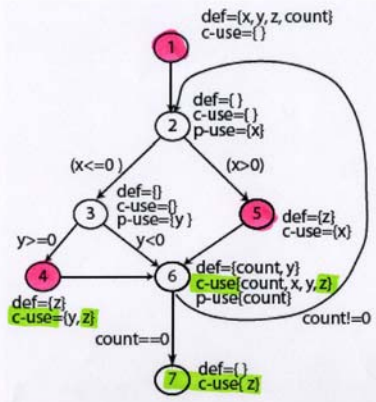


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

24

w.r.t the definition of z at ④

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

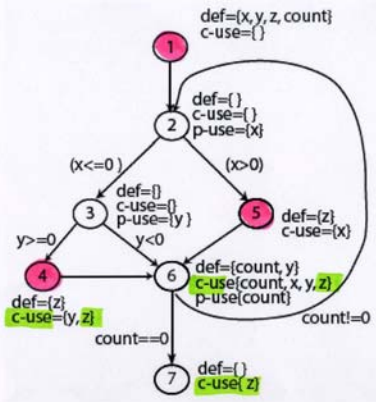


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

25

w.r.t the definition of z at ⑨

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

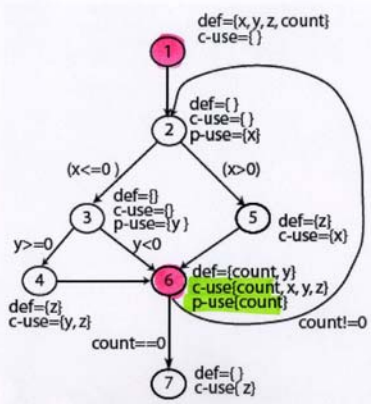


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

26

w.r.t the definition of count at ①

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```

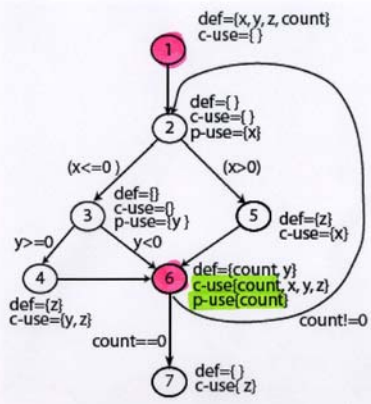


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

27

w.r.t the definitions of count at ⑥

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
```



Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

28

Exercise 3

29

Def-use pairs: Minimal set (2)

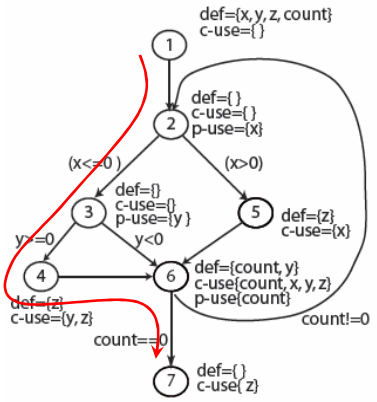
What will be also covered if we have a test case which covers $(d_1(z), u_4(z))$?

How about $(d_4(z), u_4(z))$?

30

($d_1(z)$, $u_4(z)$)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



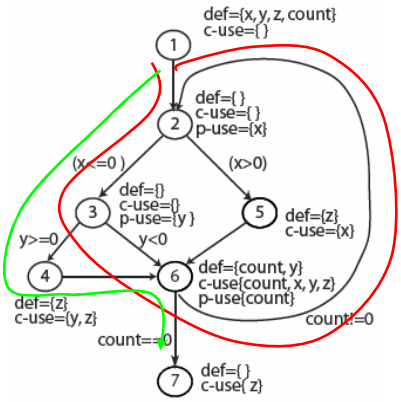
Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

1->2->3->4->6->7

31

($d_1(z)$, $u_4(z)$)

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

1->2->3->6->2->3->4->6->7

32

- Select an execution path that covers the specific def-use pair
- Identify all the def-use pairs in the selected path
- For each recognized def-use pair ($d_i(x)$, $u_j(x)$)
  - Try to find a path from node I to node j that is not def-clear path for variable x
  - If so, remove the def-use pair ($d_i(x)$, $u_j(x)$)
  - If not, keep it

33

- Consider execution path 1->2->3->4->6->7, its covered def-use pairs are
  - Variable x: (1, (2, 3)) (1, 6)
  - Variable y: (1, 4) (1, 6) (1, (3, 4))
  - Variable z: (1, 4) (4, 6) (4, 7)
  - Variable $count$: (1, 6) (6, (6, 7))
  - Def-use pairs for variable y, (1, 4) and (1, (3, 4)) can be removed because variable y is re-defined at node 6 in an alternative execution path 1->2->3->6->2->3->4->6->7

34

Exercise 4

35

C-use coverage

C-use coverage:

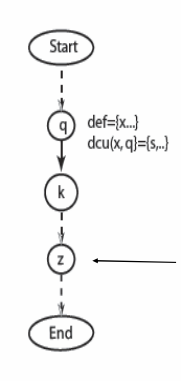
The c-use coverage of T *with respect to* (P, R) *is computed as*

$$\frac{CU_c}{(CU - CU_f)}$$

where CU_c *is the number of c-uses covered and* CU_f *the number of infeasible c-uses.* T *is considered adequate with respect to the c-use coverage criterion if its c-use coverage is 1.*

36

C-use coverage: path traversed



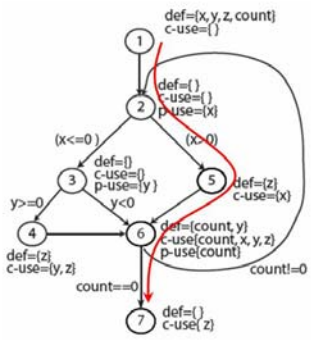
c-use of *x*

Path (Start, .. *q*, *k*, .., *z*, .. End) covers the c-use at node *z* of *x* defined at node *q* given that (*k* …, *z*) is def clear with respect to *x*

Exercise: Find the c-use coverage when program P14.16 (refer to slide 101) is executed against the following test:

t_1: <*x*=5, *y*=-1, count=1>


```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```



Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

t_1: <*x*=5, *y*=-1, *count*=1>

- covered def-c-use pairs with respect to t_1
  - Variable *x*: (1, 5) (1, 6)
  - Variable *y*: (1, 6)
  - Variable *z*: (5, 6) (5, 7)
  - Variable *count*: (1, 6)

39

Exercise 5

40

p-use coverage

P-use coverage:

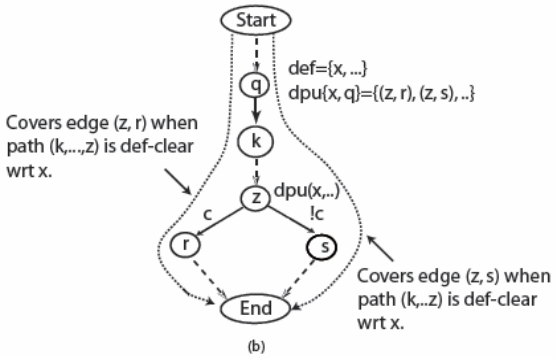
The p-use coverage of T *with respect to* (P, R) *is computed as*

$$\frac{PU_c}{(PU - PU_f)}$$

where PU_c *is the number of p-uses covered and* PU_f *the number of infeasible p-uses.* T *is considered adequate with respect to the p-use coverage criterion if its p-use coverage is 1.*

41

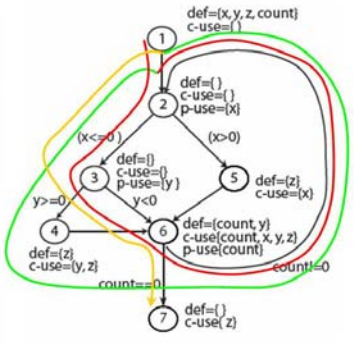
p-use coverage: paths traversed



Exercise: Find the p-use coverage when program P14.16 (refer to slide 101) is executed against test t_2: <x= -2, y= -1, count= 3>

42

```
1   begin
2     float x, y, z=0.0;
3     int count;
4     input (x, y, count);
5     do {
6       if (x≤0) {
7         if (y≥0) {
8           z=y*z+1;
9         }
10      }
11      else{
12        z=1/x;
13      }
14      y=x*y+z
15      count=count-1
16    while (count>0)
17    output (z);
18  end
```

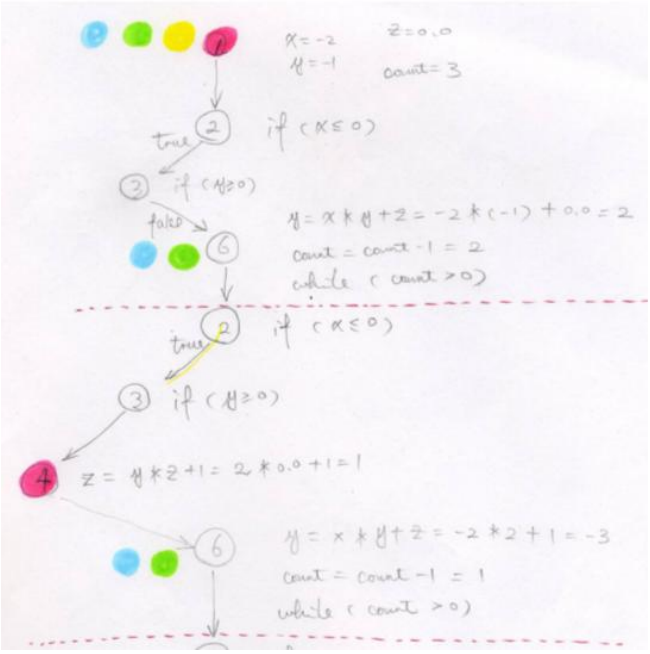


Node	Lines
1	1, 2, 3, 4
2	5, 6
3	7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18

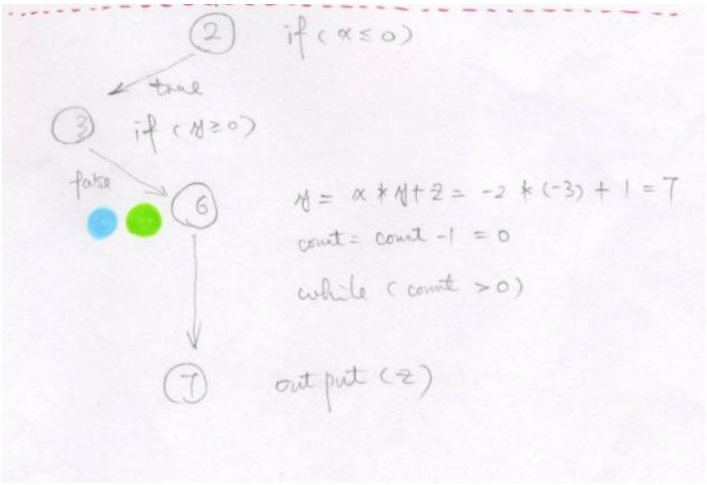
First time
Second time
Third time

t_2: <x=-2, y=-1, $count$=3>

43



44



45

- covered def-p-use pairs with respect to t_2
  - Variable *x*: (1, (2, 3))
  - Variable *y*: (1, (3, 6)) (6, (3, 4)) (6, (3, 6))
  - Variable *count*: (6, (6, 2)) (6, (6, 7))

46

Exercise 6

All-uses coverage

All-uses coverage:

The all-uses coverage of T *with respect to* (P, R) *is computed as*

$$\frac{(CU_c + PU_c)}{((CU + PU) - (CU_f + PU_f))}$$

where CU *is the total c-uses,* CU_C *is the number of c-uses covered,* PU_C *is the number of p-uses covered,* CU_f *the number of infeasible c-uses and* PU_f *the number of infeasible p-uses.* T *is considered adequate with respect to the all-uses coverage criterion if its c-use coverage is 1.*

Exercise: Is T={t_1, t_2} adequate w.r.t. to all-uses coverage for P14.16?

- covered def-use pairs with respect to t_1
  - Variable x: (1, 5) (1, 6) (1, (2, 5))
  - Variable y: (1, 6)
  - Variable z: (5, 6) (5, 7)
  - Variable *count*: (1, 6) (6, (6, 7))

49

- covered def-use pairs with respect to t_2
  - Variable x: (1, 6)
    (1, (2, 3))
  - Variable y: (1, 6) (6, 4) (6, 6)
    (1, (3, 6)) (6, (3, 4)) (6, (3, 6))
  - Variable z: (1, 4) (1, 6) (4, 6) (4, 7)
  - Variable *count*: (1, 6) (6, 6)
    (6, (6, 2)) (6, (6, 7))

50

- No, four def-use pairs are not covered with respect to t_1 and t_2, i.e.,
  - Variable y: (1, 4) and (1, (3, 4))
  - Variable z: (1, 7) and (4, 4)
- In addition, there are three infeasible def-use pairs. That is,
  - Variable z: (5, 4)
  - Variable $count$: (1, (6, 2)) (1, (6, 7))
- All use coverage is $\frac{20}{27-3} = 0.833$

51