SE 4367 Homework #13, DFG

For the following program P written in pseudo-code,

a) Draw the data flow graph for P.

b) Build the dcu/dpu table for P.

- **See Mathur's Example 7.31 (slide 34 in the lecture on data flow) as an illustration.**

```
 Program P
1)  integer X, Y, Z;
2)  input (X, Y);
3)  if (X<0 or X>8 or Y<1 or Y>3)
4)  {
5)       output ("Boundary condition failure.");
6)  } // end if invalid inputs
7)  else
8)  {
9)       Z = 0;
10)      if (X < 5)
11)      {
12)           Z = X + Y;
13)           if (Y == 1)
14)          {
15)                Z = X ^ 2;
16)          } // end if (Y==1)
17)      } // end if (X<5)
18)     else
19)     {
20)          Z = Z - X;
21)          if (Y == 0)
22)          {
23)               Z = Z * Z;
24)          } // end if (Y==2)
25)          else
26)          {
27)                Z = Z + X;
28)          } // end else !(Y==2)
29)          Z = Z + 1;
30)     } // end else !(X<5)
31)     output (X,Y,Z);
32) } // end else legal inputs
33) output ("Program ends.");
34) end;
```

Grading Rubric

13.1) Max of 50 points

- **For each block (in the CFG) incorrectly created, -5 points**
- **Incorrectly labeled nodes in the DFG, -1 point each**
- **Incorrectly labeled edges in the DFG, -1 point each**

13.2) Max of 50 points

- **For each incorrect (added or left out) variable in the dcu/dpu variable/defines pair, dcu, or dpu, -1 point each**

Formatting Submissions

In the file name, include:

- **class**
- **assignment identifier**
- **your name (or team's name)**
  - **e.g., se4367a01jdoe**

In the file (or hardcopy) submitted, include the class, assignment, and name information at the top.

Minus 5 points per violation. Potentially 30 points off for formatting mistakes!

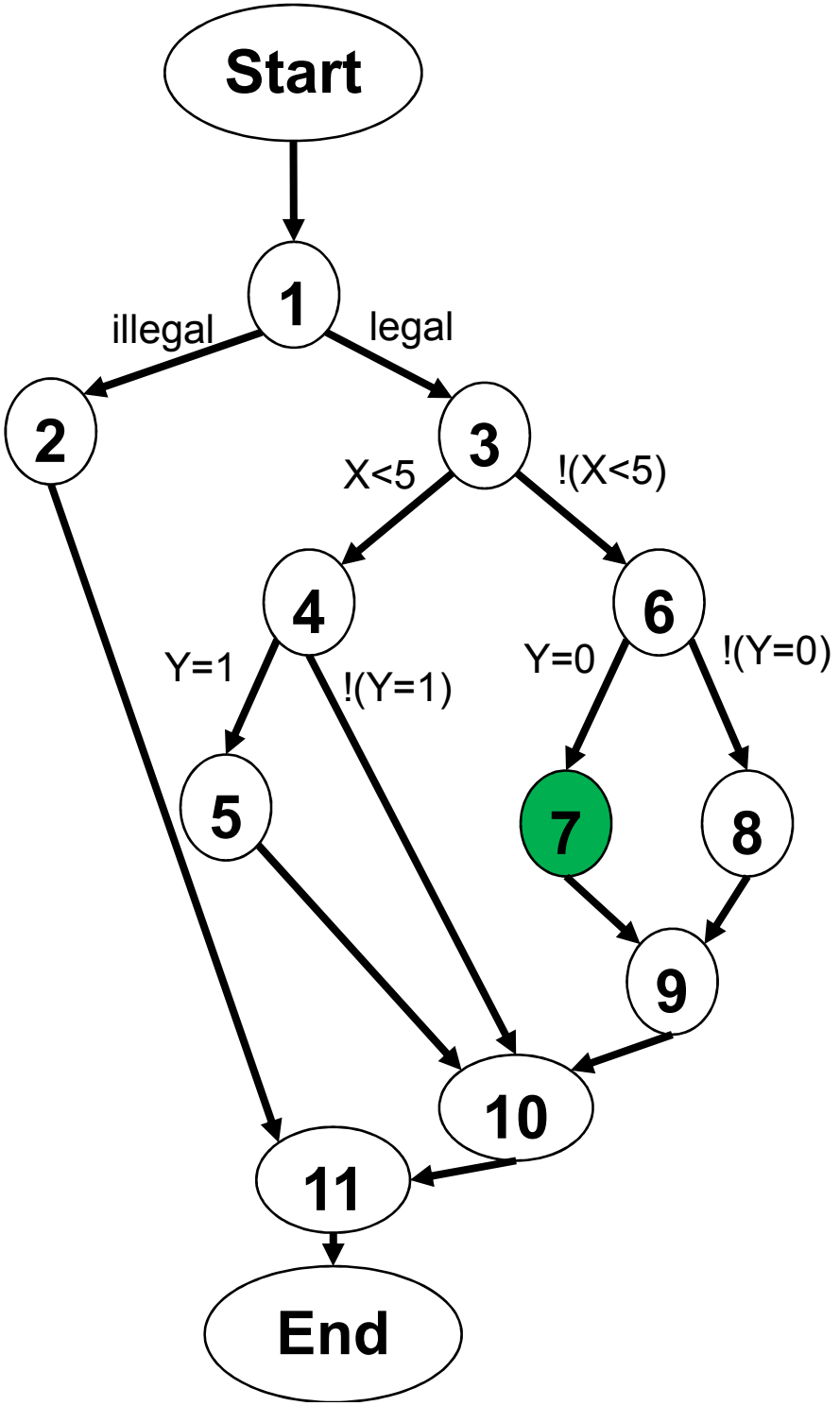
```
 Program P
1)  integer X, Y, Z;
2)  input (X, Y);
3)  if (X<0 or X>8 or Y<1 or Y>3)
4)  {
5)        output ("Boundary condition failure.");
6)  } // end if invalid inputs
7)  else
8)  {
9)        Z = 0;
10)       if (X < 5)
11)       {
12)            Z = X + Y;
13)            if (Y == 1)
14)            {
15)                 Z = X ^ 2;
16)            } // end if (Y==1)
17)       } // end if (X<5)
18)       else
19)       {
20)            Z = Z - X;
21)            if (Y == 0)
22)            {
23)                 Z = Z * Z;
24)            } // end if (Y==2)
25)            else
26)            {
27)                 Z = Z + X;
28)            } // end else !(Y==2)
29)            Z = Z + 1;
30)       } // end else !(X<5)
31)     output (X,Y,Z);
32) } // end else legal inputs
33) output ("Program ends.");
34) end;
```

Basic Blocks

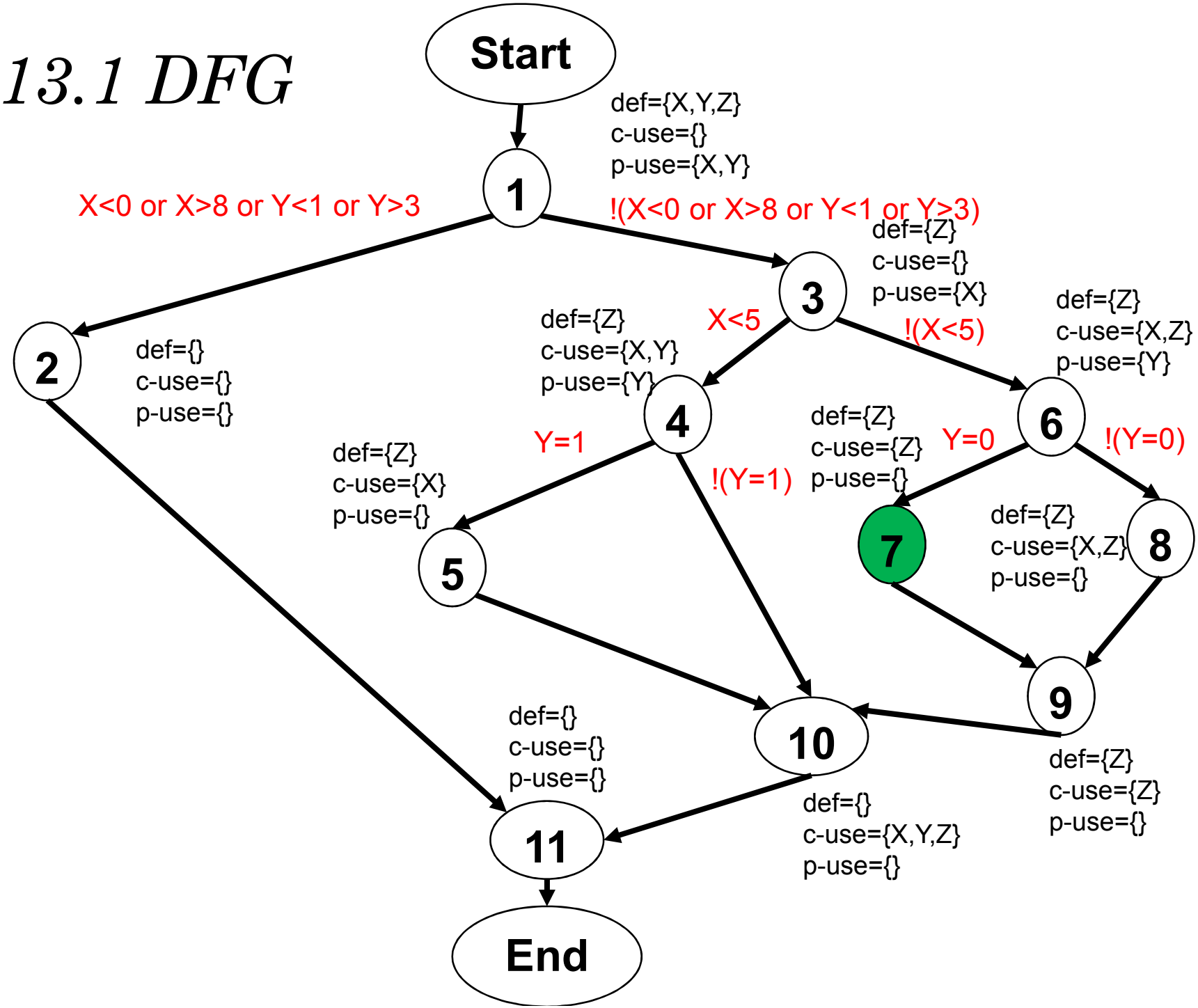
1 – 1, 2, 3 (4)
2 – 5 (6, 7, 8)
3 – 9, 10 (11)
4 – 12, 13 (14)
5 – 15 (16, 17, 18, 19)
6 – 20, 21 (22)
7 – 23 (24, 25, 26)
8 – 27 (28)
9 – 29 (30)
10 – 31 (32)
11 – 33 (34)

CFG

Block	LOC
1	1,2,3
2	5
3	9,10
4	12,13
5	15
6	20,21
7	23 (infeasible)
8	27
9	29
10	31
11	33



13.1 DFG


Start
def={X,Y,Z}
c-use={}
p-use={X,Y}
1
X<0 or X>8 or Y<1 or Y>3
!(X<0 or X>8 or Y<1 or Y>3)
2
def={}
c-use={}
p-use={}
3
def={Z}
c-use={}
p-use={X}
X<5
!(X<5)
4
def={Z}
c-use={X,Y}
p-use={Y}
6
def={Z}
c-use={X,Z}
p-use={Y}
Y=1
!(Y=1)
5
def={Z}
c-use={X}
p-use={}
Y=0
!(Y=0)
7
def={Z}
c-use={Z}
p-use={}
8
def={Z}
c-use={X,Z}
p-use={}
9
def={Z}
c-use={Z}
p-use={}
10
def={}
c-use={}
p-use={}
11
def={}
c-use={X,Y,Z}
p-use={}
End

Creating the dcu/dpu Table

Note that it is helpful to have the DFG to draw the dcu/dpu table.

It may also be useful to have the def-use table, which has the same definition and usage information, but cannot be used to identify the def-clear paths.

Def-Use Table

Node	def	c-use	p-use
1	{X,Y,Z}	{}	{X,Y}
2	{}	{}	{}
3	{Z}	{}	{X}
4	{Z}	{X,Y}	{Y}
5	{Z}	{X}	{}
6	{Z}	{X,Z}	{Y}
7	{Z}	{Z}	{}
8	{Z}	{X,Z}	{}
9	{Z}	{Z}	{}
10	{}	{X,Y,Z}	{}
11	{}	{}	{}

13.2 dcu/dpu Table

Variable (v)	Defined at node (n)	dcu (v,n)	dpu (v,n)
X	1	{4,5,6,8,10}	{(1,2),(1,3), (3,4),(3,6)}
Y	1	{4,10}	{(1,2),(1,3), (4,5),(4,10), (6,7),(6,8)}
Z	1	{}	
Z	3	{6}	
Z	4	{10}	
Z	5	{10}	
Z	6	{6,7,8}	
Z	7	{7,9}	
Z	8	{8,9}	
Z	9	{9,10}	