Final Review

PLT-4115

Q1. Consider the basic block:

```
y := 3
x := y
z := 4 * x
```

Now consider the local optimizations:

- constant propagation
- copy propagation,
- constant folding.

- For this example, what is the best order in which to apply the three optimizations, if each can be applied only once?

  Ans: copy propagation, constant propagation, constant folding correct

Q2. Consider the basic block:

```
y := 3
x := y
z := 4 * x
```

Now consider the local optimizations:

- constant propagation
- copy propagation,
- constant folding.

- For this example, What is the worst possible order (i.e., requires the most passes) for the basic block?

Ans: constant folding, constant propagation, copy propagation

Q3. Consider the following intermediate code:

1. x := 5
2. if y >1 goto Label3
3. Label1:
4. w := w + 1
5. if y > 2 goto Label3
6. Label2:
7. q := 3
8. if z < 1 goto Label1
9. Label3:
10. w := 2
11. if z > 1 goto Label2
12. q := y + w

a. Draw the CFG where each node is a BB.

b. Which variables are live immediately before the execution of statement 7? Assume only variable q is live after the statement in line 12.

Ans: y,z,w

Q3. Consider the following intermediate code:

1. x := 5
2. if y >1 goto Label3
3. Label1:
4. w := w + 1
5. if y > 2 goto Label3
6. Label2:
7. q := 3
8. if z < 1 goto Label1
9. Label3:
10. w := 2
11. if z > 1 goto Label2
12. 12: q := y + w

c. Assume the constant propagation algorithm has completed. Which of the following statements is true?

- L_N is the statement at line N
- C(L,v,in) = C means that at the "in" of statement L variable v is some constant
- C(L,v,in) = ⊤ means v is not a constant.

C(L7,	w,	in)	=	T
C(L2,	y,	out)	=	C
C(L5,	x,	out)	=	C
C(L4,	y,	in)	=	T
C(L8,	z,	out)	=	C

Q3. Consider the following intermediate code:

1. x := 5
2. if y >1 goto Label3
3. Label1:
4. w := w + 1
5. if y > 2 goto Label3
6. Label2:
7. q := 3
8. if z < 1 goto Label1
9. Label3:
10. w := 2
11. if z > 1 goto Label2
12. 12: q := y + w

c. Assume the constant propagation algorithm has completed. Which of the following statements is true?

- L_N is the statement at line N
- C(L,v,in) = C means that at the "in" of statement L variable v is some constant
- C(L,v,in) = ⊤ means v is not a constant.

C(L7,	w,	in)	=	T
C(L2,	y,	out)	=	C
C(L5,	x,	out)	=	C
C(L4,	y,	in)	=	T
C(L8,	z,	out)	=	C

Q4. Consider the following intermediate code:

1. x := 5, z:= 2, y:= 3
2. if y >1 goto Label3
3. Label1:
4. w := w + 1
5. if y > 2 goto Label3
6. Label2:
7. q := 3
8. if z < 1 goto Label1
9. Label3:
10. w := 2
11. if z > 1 goto Label2
12. 12: q := y + w

a. Which lines (using the numbering given above) are now unreachable?

Do constant propagation and dead code elimination

1. x := 5, z:= 2, y:= 3
2. if y >1 goto Label3
3. Label1:
4. w := w + 1
5. if y > 2 goto Label3
6. Label2:
7. q := 3
8. if z < 1 goto Label1
9. Label3:
10. w := 2
11. if z > 1 goto Label2
12. q := y + w

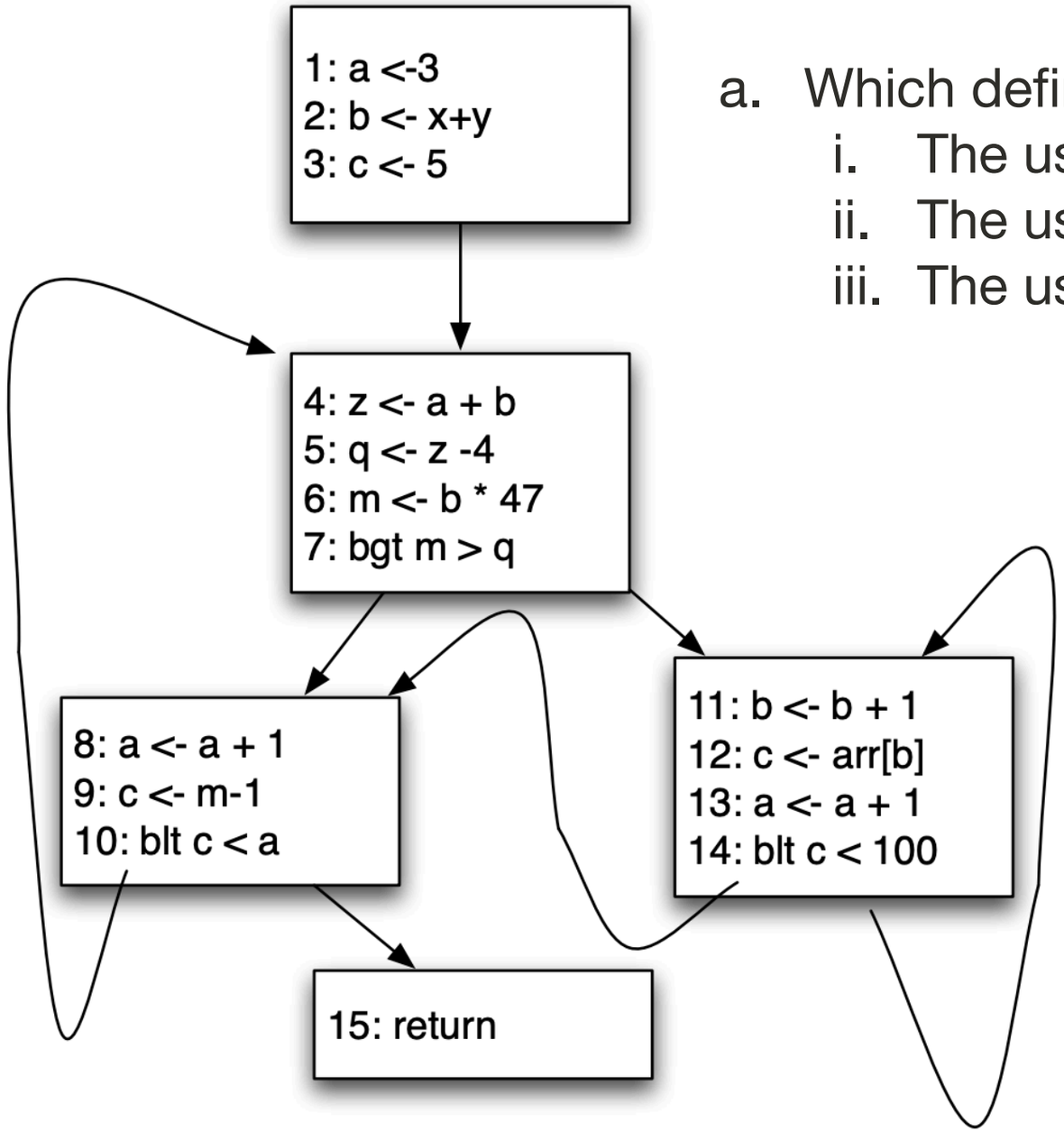
Q5. Optimize the following intermediate code:

1: z := 3
2: if b > 0 goto Label1
3: x := 1
4: y := 2
5: z := x + y
6: goto Label2
7: Label1:
8: w := x + 1
9: y := x + 1
10: Label2:
11: a := x + y
12: b := a * z

1: z := 3
2: if b > 0 goto Label1
3: x := 1
4: y := 2
5: z := ~~x + y~~ 3
6: goto Label2
7: Label1:
8: w := x + 1
9: y := ~~x + 1~~ w
10: Label2:
11: a := x + y
12: b := a * ~~z~~ 3

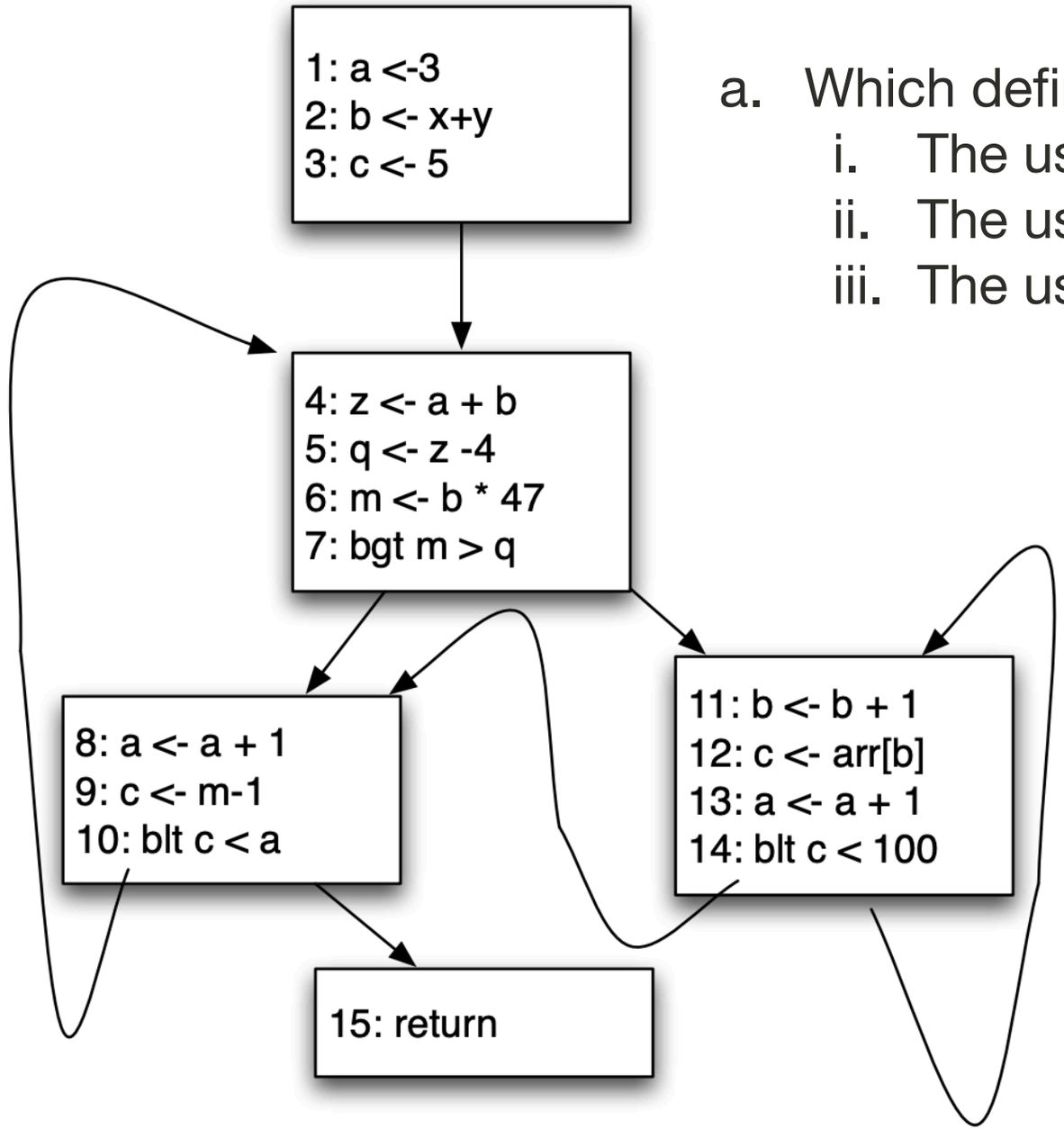
Line 8 can also be removed if you assume w will not be used after line 12

Q6. Consider the following CFG



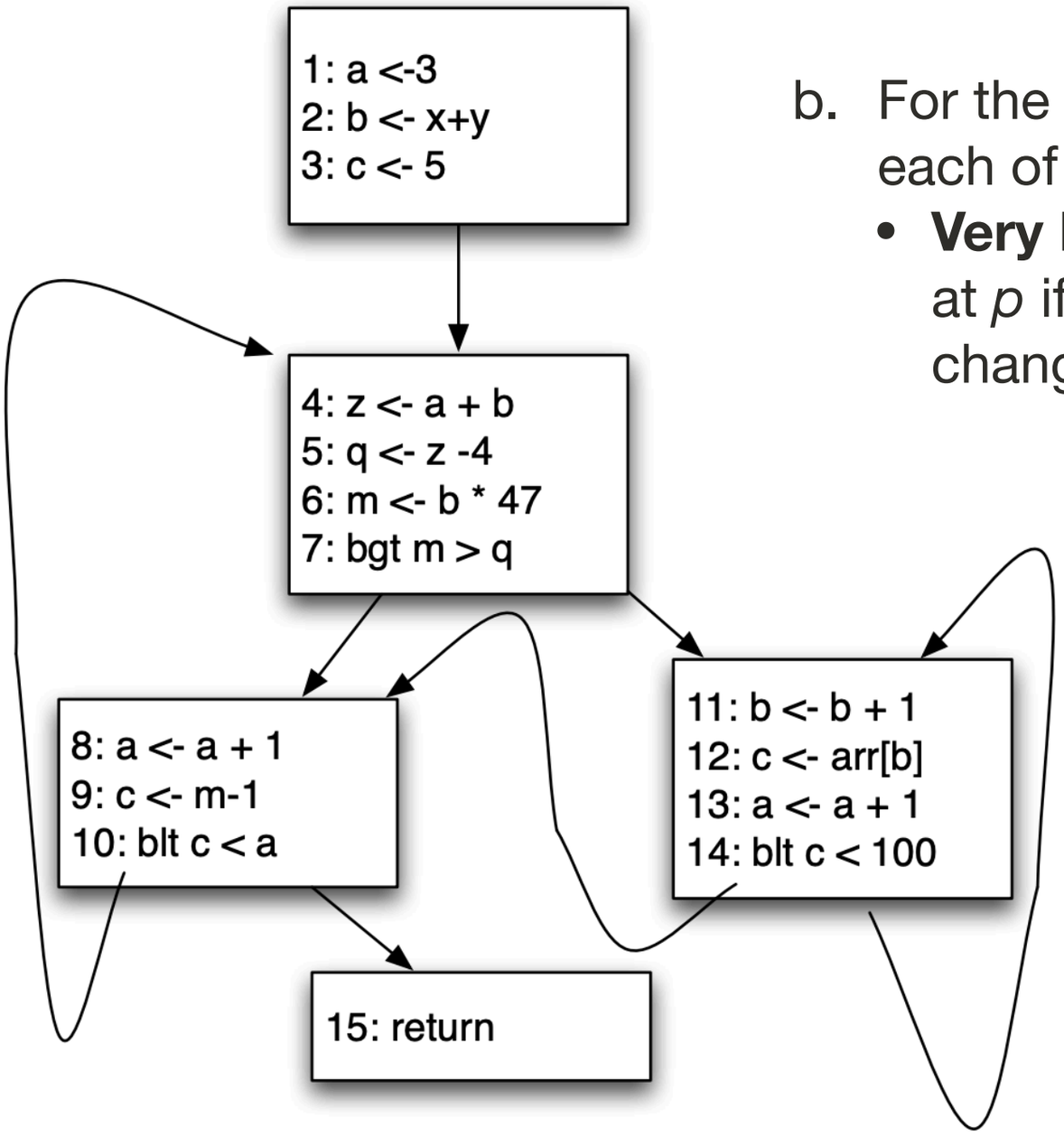
a. Which definitions reach the following uses:
   i. The use of a in instruction 4.
   ii. The use of a in instruction 8.
   iii. The use of b in instruction 6.

Q6. Consider the following CFG



a. Which definitions reach the following uses:
   i. The use of a in instruction 4. 1,8
   ii. The use of a in instruction 8. 1,8,13
   iii. The use of b in instruction 6. 2,11

Q6. Consider the following CFG



b. For the same program fragment, indicate whether each of the following expressions is “very busy”
   - **Very Busy Expressions**: An expression is very busy at *p* if it is evaluated on *every* path *from p* before it changes in value. (**Backward Must**)

	a + 1	m - 1	a + b	b * 47	x + y	b+1	arr[b]
3	Y	N	Y	Y	N	N	N
7	Y	Y	N	N	N	N	N
10	N	N	N	N	N	N	N
14	Y	Y	N	N	N	N	N
15	N	N	N	N	N	N	N