HA-3

1)

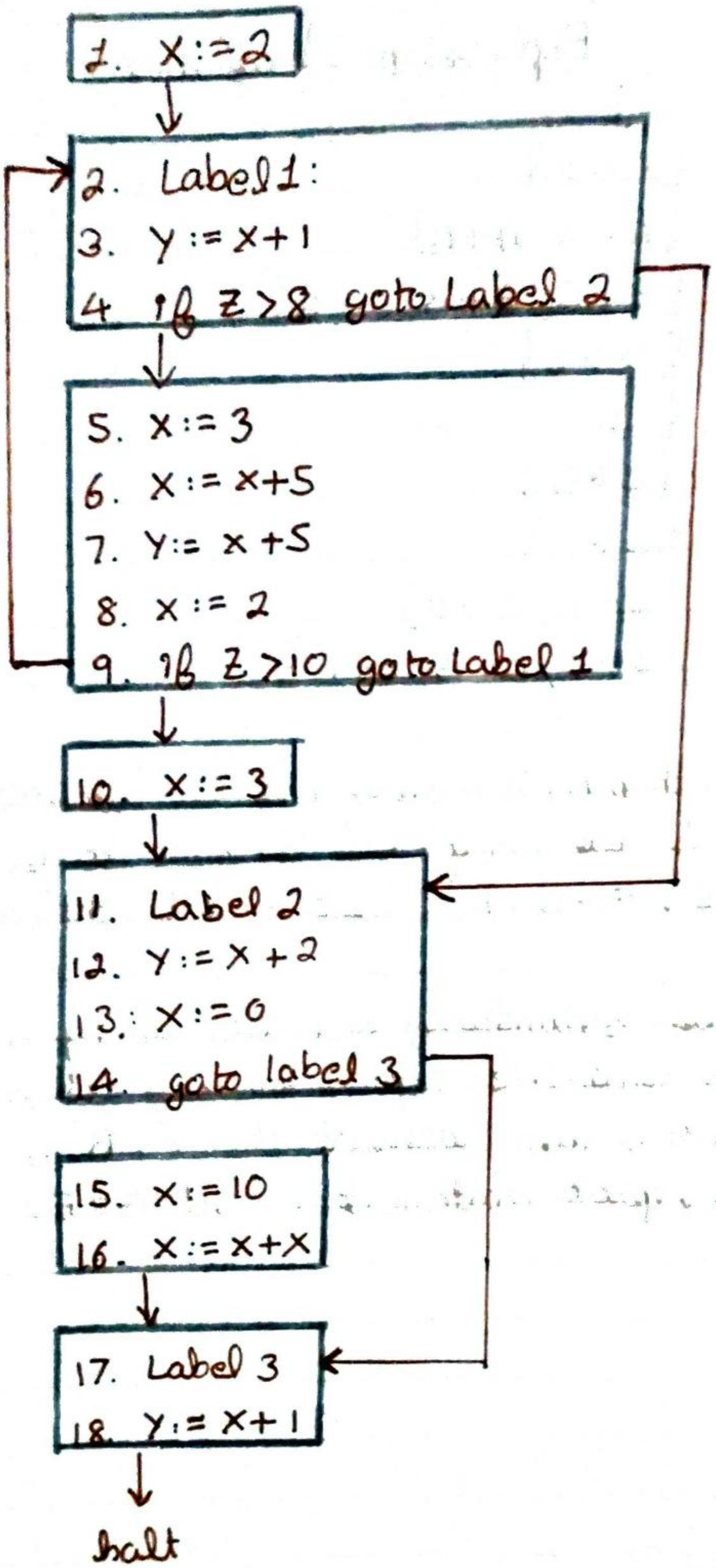
a)

Statement	Expressions Available
1. A = 7;	{ }
2. B = A + 2;	{A + 2}
3. C = A + B;	{A + 2, A + B}
4. D = C + B;	{A + 2, A + B, C + B}
5. B = D; / B = C + B;	{A + 2}
6. A = A + B;	{ }
7. E = C + D;	{C + D}
8. F = E; / F = C + D;	{C + D}
9. G = C + B;	{C + D, C + B}
10. H = E + F;	{C + D, C + B, E + F}

b) If E & C were aliased, then C + D would not be available after statement 7 (E = C + D). & we would not be able to perform CSE on statement 8 (F = E / F = C + D). Root will be the same.

c) Aliasing might increase redundancy because an expression that looks different than an available expression may actually be redundant. ~~Root~~ If B & D were aliased, then C + B is the same as C + D, & we could replace statement 9 with G = E.

2)



Leaders => {1, 2, 5, 10, 11, 15, 17}

3) Code after constant propagation 1st time =>

1. x := 2
2. Label 1
3. y := 2 + 1
4. if z > 8 goto label 2
5. x := 3
6. x := 3 + 5
7. y := x + 5
8. x := 2
9. if z > 10 goto label 1
10. x := 3
11. Label 2
12. y := x + 2
13. x := 0
14. goto Label 3
15. x := 10
16. x := 10 + 10
17. Label 3
18. y := 0 + 1

Code after 2nd constant propagation =>

1. x := 2
2. Label 1
3. y = 3
4. if z > 8 goto Label 2
5. x := 3
6. x := 8
7. y := 8 + 5
8. x := 2
9. if z > 10 goto Label 1
10. x := 3
11. Label 2
12. y := x + 2
13. x := 0
14. goto Label 3
15. x := 10
16. x := 20
17. Label 3
18. y := 1

Code after 1st constant folding =>

1. X := 2
2. Label 1
3. Y := 3
4. if Z > 8 goto Label 2
5. X := 3
6. X := 8
7. Y := X+5
8. X := 2
9. if Z > 10 goto Label 1
10. X := 3
11. Label 2
12. Y := X+2
13. X := 0
14. goto Label 3
15. X := 10
16. X := 20
17. Label 3
18. Y := 1

Code after 2nd constant folding =>

1. X := 2
2. Label 1
3. Y := 3
4. if Z > 8 goto Label 2
5. x := 3
6. X := 8
7. Y := 13
8. X := 2
9. if Z > 10 goto Label 1
10. X := 3
11. Label 2
12. Y := X+2
13. X := 0
14. goto Label 3
15. X := 10
16. X := 20
17. Label 3
18. Y := 1

4) a) Loop Def = {z, y, x}
   Loop Use = {x, z, c, y}
   } Variables

Invariant variables = loop use - Loop Def
= {x, z, c, y} - {z, y, x}
= {c}

Invariant expression => [redacted] in line S. => 2 * c

Yes, it can be factored out of the loop body.

[redacted] = [redacted] temp = 2 * c

[redacted] z = temp

But if the loop is never executed, then factoring it out might become overhead.

b) Basic induction variables {I = I ± S} ∴ Here 7| x := x + 1 is III
= {[redacted] x}

Mutual induction variables {J = I * [redacted] ± D} C, D are loop invariants
= {y} ∴ Here 6. | y = z * x | is III

c) After code motion :=>

1. x = a * a
2. c = -1 * x
3. z = 2 * c
4. L1 :
5. if (x >= 100) goto L2
6. y = z * x
7. x = x + 1
8. goto L1
9. L2 :

4 c) After strength reduction =>

1. x = a * a
2. c = -1 * x
3. z = 2 * c
4. t1 = z * x
5. t2 = z * 100
6. L1 :
7. if (t1 >= t2) goto L2
8. y = t1
9. t1 = t1 + z
10. goto L1
11. L2

5)

Block	LiveIn	LiveOut
b1	{y, z, w}	{y, z, w}
b2	{y, z, w}	{y, z, w}
b3	{y, z, w}	{y, z, w}
b4	{y, z}	{w, y, z}
b5	{w, y}	{}