Sample Solution

1. Identify all dependences in the following examples. Write down the element and the two statements on which the dependence occurs, and identify the type of the dependence.

a)
```
S1 voltage = 220;
S2 ampere = 3;
S3 watts = voltage * ampere;
```

S1 δ S3 on `voltage` S2 δ S3 on `ampere`

b)
```
S1 a = 0;
S2 if (b != 0) {
S3   a = c;
S4 } else {
S5   a = c / b;
S6 }
S7 print(a);
```

list of dependences when considering reaching definitions:

S1 δ^0 S3, S5 on a S3 δ S7 on a S5 δ S7 on a

c)
```
S1 n = 0;
S2 a[n] = 0;
S3 n++;
S4 a[n] = 1;
S5 n++;
S6 a[n] = a[n-2] + a[n-1];
```

list of dependences when considering reaching definitions:

on n:	S1 δ S2, S3	S1 δ^0 S3	
	S2 δ^{-1} S3		
	S3 δ S4, S5	S3 δ^{-1} S3	S3 δ^0 S5
	S4 δ^{-1} S5		
	S5 δ S6	S5 δ^{-1} S5	
on a[0]:	S2 δ S6		
on a[1]:	S4 δ S6		

complete list of dependences:

S1 δ S3,S4,S5,S6 on n	S1 δ^0 S3,S5 on n	
S2 δ^{-1} S3,S5 on n	S2 δ S6 on a[0]	
S3 δ S4,S5,S6 on n	S3 δ^{-1} S3, S5 on n	S3 δ^0 S5 on n
S4 δ^{-1} S5 on n	S4 δ S6 on a[1]	
S5 δ S6 on n	S5 δ^{-1} S5 on n	

2. Write down the first and the last iteration vector of the iteration space of the following loop nest. How many elements does the iteration space contain?

```
    for (i=1; i<10; i++) {
      for (j=1; j<=20; j++) {
        for (k=-1; k>-10; k--) {
          for (l=0; l<10; l=l+2) {
S1          A[i+j, k+3-l] = B[-k, i, j] * C[i,j,l];
          }
        }
      }
    }
```

first iteration vector $\boldsymbol{i}_l = (1, 1, -1, 0)$
last iteration vector $\boldsymbol{i}_{last} = (9, 20, -9, 8)$
elements in the iteration space: 9 * 20 * 9 * 5 = 8100

3. Which one of the following iteration vectors for the above loop nest is smaller?
a) (4, 3, -2, 4) < (4, 4, -3, 2)
b) (5, 6, -4, 0) > (5, 6, -3, 8)

4. Suppose we have normalized the iteration numbers in the iteration vectors. Which array elements are then accessed in statement S1 of the above loop nest for the loop iteration vector $\boldsymbol{i} = (1, 2, 3, 4)$?

i = 1, j = 2, k = -3, l = 6 $\rightarrow$ A[3, -6] = B[3, 1, 2] * C[1, 2, 6]

(if you assume that $\boldsymbol{i} = (0, 0, 0, 0)$ is the first iteration vector, then we get
i = 2, j = 3, k = -4, l = 8 $\rightarrow$ A[5, -9] = B[4, 2, 3] * C[2, 3, 8]

5. For the following loop nest

```
    for (i=1; i<=N; i++) {
      for (j=1; j<=M; j++) {
        for (k=1; k<=L; k++) {
S1        A[i+3, j-1, k] = A[i, j+1, k+1] + 7;
        }
      }
    }
```

write down two concrete iteration vectors that cause a true dependence in statement S1.

The distance vector is $\boldsymbol{d} = (3, -2, -1)$, so any two iteration vectors within the iteration space with difference $\boldsymbol{d}$ form a solution.
For example, given $\boldsymbol{i}_l = (5, 5, 5)$, then $\boldsymbol{i}_2 = (8, 3, 4)$ (both access A[8, 4, 5]).

6. What do the distance and the direction vectors look like for the iteration vectors identified in the previous problem?

The distance vector is $\boldsymbol{d} = (3, -2, -1)$, and the direction vector is $\boldsymbol{D} = (<, >, >)$

7. For the following loop, write down S1 and S2 such that they form a loop-carried forward true dependence.

```
    for (i=0; i<N; i++) {
S1
S2
    }
```

The obvious solution is

```
S1    A[i+1] = …
S2    … = A[i]
```

or

```
S1    … = A[i]
S2    A[i+1] = …
```

in the two solutions above, not only “1” but any positive integer constant > 0 and $< N$ is correct.

Note that a simple scalar also forms a loop-carried forward true dependence:

```
S1    a = …
S2    … = a
```

and also

```
S1    … = a
S2    a = …
```

8. Draw the dependence graph for the following loop (identify loop-carried as well as loop-independent dependences and identify the level of the dependence)

```
    for (i=1; i<=100; i++) {
S1    A[i, N] = D[i-1, 0];
      for (j=1; j<=100; j++) {
S2      B[j-1] = C[j, N];
        for (k=1; k<=100; k++) {
S3        C[j+1, k] = B[j-1] + D[i, k];
        }
S4      D[i, j] = A[i+j, N] + 2;
      }
    }
```

1. <u>S1 → S4 on A</u>
<i, i+j>, <N, N>

<N, N>: always match
<i, i+j>: $i = i' + j \rightarrow i = i + \Delta i + j \rightarrow \Delta i = -j$
since $1 \leq j \leq 100$, Δi is always negative → $\boldsymbol{D} = (>)$

→ level-1 antidependence S4 → S1

2. <u>S2 → S3 on B</u>
<j-1, j-1>

enclosed in level-2 loop, but no level-1 index → $D_1 = (*)$
<j-1, j-1>: $j - 1 = j' - 1 \rightarrow j - 1 = j + \Delta j - 1 \rightarrow \Delta j = 0 \rightarrow D_2 = (=)$
$\boldsymbol{D} = (*, =) = \{ (<, =), (=, =), (>, =) \}$

→ (<, =): level-1 true dependence
→ (=, =): loop-independent true dependence
→ (>, =): level-1 antidependence S3 → S2

3. S3 → S2 on C

   <j+1, j>, <k, N>

   <k, N>: dependence assumed to exist since no further information about N is available
   <j+1, j>: j + 1 = j’ → j + 1 = j + Δj → Δj = 1 → D_2 = (<)
   enclosed in level-2 loop, but no level-1 index → D_1 = (*)
   D = (*, <) = { (<, <), (=, <), (>, <) }

   → (<, <): level-1 true dependence
   → (=, <): level-2 true dependence
   → (>, <): level-1 antidependence S2 → S3

4. S4 → S3 on D

   <i, i>, <j, k>

   <j, k>: all directions are possible (j < k, j = k, j > k), hence D_2 = (*)
   <i, i>: obviously D_1 = (=)
   D = (=, *) = { (=, <), (=, =), (=, >) }

   → (=, <): level-2 true dependence
   → (=, =): loop-independent true dependence
   → (=, >): level-2 antidependence S3 → S4

5. S4 → S1 on D

   <i, i-1>, <j, 0>

   <j, 0>: no dependence possible (no need to test <i, i-1>)
   → no dependences

6. S1 → S1 (output dependence test for A)

   <i, i>, <N, N>

   obviously ***D*** = (=, =), hence no loop-carried output dependence

7. S2 → S2 (output dependence test for B)

   <j-1, j-1>

   enclosed in level-2 loop, but no level-1 index → D1 = (*)
   → ***D*** = (*, =): level-1 output dependence

8. S3 → S3 (output dependence test for C)

   <j+1, j+1>, <k, k>

   again, the LHS has one less dimension than the number of loops containing the statement
   → ***D*** = (*, =, =): level-1 output dependence

9. <u>S4 → S4 (output dependence test for D)</u>

<i, i>, <j, j>

→ ***D*** = (=, =): no loop-carried output dependence