Jerry Perez
CS4515-Homework 5

4.14

```
for (i=0;i <100;i++) {
 A[i] = B[2*i+4];
 B[4*i+5] = A[i];
}
```

a. to check for loop-carried dependency existence, we can use the GCD (Greatest Common Devisor) test. A depency exists if GCD(2,4) divides 5-4.

↳ GCD(2,4) = 2

↳ 5 - 4 mod 2 = 1

→ GCD(2,4) does NOT divide 5-4 which tells us there is no loop carried dependency for B[].

→ Since the same index is used for A[] in the loop body, no loop carried dependencies exist for A[] also.

b.

```
for (i=0;i <100;i++) {
A[i] = A[i] * B[i]; /* S1 */

B[i] = A[i] + c; /* S2 */
A[i] = C[i] * c; /* S3 */
C[i] = D[i] * A[i]; /* S4 */
```

- There is a true dependency over variable A in statement 1 and statement 2. There is another ocurrence over A in statement 3 and 4. Finally, there is a loop carried dependency over A between statement 1 and 4

- There is an output dependency between statement 1 and 3 since both use instance variable A.

- There is an antidependency over B between statements 1 and 2 since statement 2 calculates what is used by S1. There is another antidependency over A between statements 2 and 3, and another over C between statements 3 and 4 for the same reason identified for the first antidependency.

- Both output and anti dependencies can be avoided by renaming the violating variables. A on the left can be changed to A1, B on the left of statement 2 to B1, and C on statement 4 to C1. Doing this will avoid all output and antidependencies.

c.

```
for (i=0;i<100;i++) {
 A[i] = A[i] + B[i]; /* S1 */
 B[i+1] = C[i] + D[i]; /* S2 */
}
```

- there is an antidependence over B between iteration i and $i+1$.

- this loop is not parallel but can be made parallel by doing something along the lines of:

```
A[0] = A[0] + B[0];
for (i=0; i<99; i++)
{
    B[i+1] = C[i] + D[i];
    A[i+1] = A[i+1] + B[i+1];
}
B[100] = C[99] + D[99];
```