- Investigate the given code excerpt. Can this code be safely parallelized manually? Can this code be safely parallelized by the compiler?

```
void copy(double* x, double* y) {
    for(int i = 0; i < 1024; i++) {
        x[i] = y[i];
    }
}
```

This Code cannot be parrallelized because x and y may be aliased. The Code could in theory be safely parrallelized if an Aliasing Check is introduced, but probably neither a Programmer nor the Compiler would do this.

- Normalize the following loop nest:

```
for (int i=4; i<=N; i+=9) {
    for (int j=0; j<=N; j+=5) {
        A[i] = 0;
    }
}
```

```
for (int i=0; i<=((N-4)/9); i++) {
    for (int j=0; j<=(N/5); j++) {
        A[4 + i * 9] = 0;
    }
}
```

=> Since i and j are declared in the for-loop they do not need to be set to their final Value after the Loop.

- Does the following code excerpt hold any dependencies? If not, how would you parallelize it? If yes, what are the distance and direction vectors?

```
for(int i = 1; i < N; i++) {
    for(int j = 1; j < M; j++) {
        for(int k = 1; k < L; k++) {
            a[i+1][j][k-1] = a[i][j][k] + 5;
        }
    }
}
```

There is a Loop carried True dependency on the outermost Loop (assuming N > 1 and M > 1 and L > 1). This is because the value at a[i+1][j][k-1] is written in the corresponding Loop Iteration

and since i is then incremented the same Location is read by a[i][j][k].

This Dependency does not apply to all Values in the Array:

- Values with a k-Index of 0 are never read.
- Values with a k-Index of L are never written.

The Distance is (1,0,-1).
The Direction Vector is (>,=,<).