Exercise 1

Team: Summit

Consider the following individual code snippets, and analyze them regarding dependencies.

Regarding each snippet

- What are the data dependencies?

- Parallelize and optimize the code

Snippet 1:

```
for (int i = 0; i < n - 1; i++)
{
    x[i] = (y[i] + x[i + 1]) / 7;
}
```

Anti - dependence, the location in memory is read before that same location is written to.
Paralyzed: by privatization

```
#pragma omp parallel for
    for (int i = 0; i < n - 1; i++)
    {
        z[i] = x[i];
    }
#pragma omp parallel for
    for (int i = 0; i < n - 1; i++)
    {
        x[i] = (y[i] + z[i + 1]) / 7;
    }
```

Snippet 2:

```
for (int i = 0; i < n; i++)
{
    a = (x[i] + y[i]) / (i + 1);
    z[i] = a;
}

double f = sqrt(a + k);
```

True - dependence, memory is written to before it is read.

Paralyzed by eliminating a[n], and use of #pragma.

```
#pragma omp parallel for
    for (int i = 0; i < n; i++)
    {
        z[i] = (x[i] + y[i]) / (i + 1);
    }
    //f = sqrt(a + k) -> was never used
```

Snippet 3:

```
for (int i = 0; i < n; i++)
{
    x[i] = y[i] * 2 + b * i;
}

for (int i = 0; i < n; i++)
{
    y[i] = x[i] + a / (i + 1);
}
```

True - dependence, memory is written to before it is read, but there're its own loops.

Paralyzed by use of #pragma.

```
#pragma omp parallel for
    for (int i = 0; i < n; i++)
    {
        x[i] = y[i] * 2 + b * i;
    }
#pragma omp parallel for
    for (int i = 0; i < n; i++)
    {
        y[i] = x[i] + a / (i + 1);
    }
```