Exercise 2

Snippet 1

double factor = 1;

for (int i=0; i < n; i++) {

x[i] = factor \* y[i];

factor = factor / 2;

}

-loop independent dependency

-the factor depends on the iteration, so can be solved with easy maths

Snippet 2  
  
for (int i = 1; i<n; i++) {

x[i] = (x[i] + y[i-1]) / 2;

y[i] = y[i] + z[i] \* 3;

}

-true dependency RAW for y, can’t be solved easily; y[i-1] depends on y[i] of previous loop

Snippet 3  
  
x[0] = x[0] + 5 \* y[0];

for (int i = 1; i<n; i++) {

x[i] = x[i] + 5 \* y[i];

if ( twice ) {

x[i-1] = 2 \* x[i-1]

}

}

-here we have a flag twice, which if set introduces a loop-carried true dependency and anti-dependency because x[i-1] of x[i-1] = 2 \* x[i-1] are read and written to in the previous iteration with x[i] = x[i] + 5 \* y[i]; true dependencies can’t be resolved

compiled with O1  
  
  
Exercise 1:

Snippet 1:

Unparallelized execution: 0.020172

parallelized execution: 0.077661

Snippet 2:

unparallelized execution: 0.026287

Can't be parallelized!

Snippet 3:

Unparallelized execution, flag set: 0.014079

Unparallelized execution, flag not set: 0.010923

program can't be parallelized if flag set.

parallelized execution, flag not set: 0.006296

compiled with O3

Exercise 1:

Snippet 1:

Unparallelized execution: 0.020238

parallelized execution: 0.053107

Snippet 2:

unparallelized execution: 0.020771

Can't be parallelized!

Snippet 3:

Unparallelized execution, flag set: 0.017326

Unparallelized execution, flag not set: 0.008177

program can't be parallelized if flag set.

parallelized execution, flag not set: 0.006416