1 Dependency Analysis: Loop Dependencies

Dependencies throughout different iterations of loops are difficult to detect. Here, the so-called distance and direction vector help to distinguish between loop-carried and loop-independent dependencies. Analyse the following code fragment and identify all occurring dependencies.

```plaintext
for i ← 1 to N-1 do
  for j ← 1 to N-1 do
    for k ← 1 to N-3 do
      A(i+1, j+1, k) ← A(i, j, k) + A(i, j+1, k+3)
    od
  od
od
```

Are the above dependencies to be solved? Justify your solution!

2 Parallel Computation of π

With
\[
\phi(x) = \frac{1}{1 + x^2} \quad \text{and} \quad \int \phi(x)dx = \arctan(x)
\]

one could compute π via the integration of φ(x) over [0, 1]. The following code fragment shows how to compute π sequentially, subdividing the unit interval into N stripes.

```plaintext
int: i, N
double: h, x, sum, PI

h ← 1/N
sum ← 0

for i ← 1 to N do
  x ← h*(i - 0.5)
  sum ← sum + 4/(1 + x*x)
  od

PI ← h*sum
```

Extend the program with valid OpenMP directives to compute π in parallel and think about sufficient synchronisation of the threads!
Parallel Min-Max-Search

To find the minimal and maximal elements $min$ and $max$, resp., of a 3-dimensional integer array $A$ of size $3 \times 1000 \times 1000$, the following sequential code is used.

```c
int i, j, k, A, min, max

min ← A[1][1][1]   // minimal element
max ← A[1][1][1]   // maximal element

for i ← 1 to 3 do
    for j ← 1 to 1000 do
        for k ← 1 to 1000 do
            if A[i][j][k] < min then min ← A[i][j][k] fi
            if A[i][j][k] > max then max ← A[i][j][k] fi
        od
    od
od
```

Extend the program with valid OpenMP directives for the parallelisation of one loop and – if necessary – think about sufficient synchronisation of the threads!