HOMEWORK: COMPILER AUTO-PARALLELISATION (30 POINTS)

Study the auto-parallelisation options of the Intel and PGI compilers:

- Intel: https://software.intel.com/sites/default/files/m/d/4/1/d/8/4-1-ProgTools_-_Automatic_Parallelization_with_Intel_C2_AE_Compilers.pdf;
- PGI: http://www.pgroup.com/doc/pgiug.pdf, Section 3.7.

Consider the following codes:

1.
```
for(i = 0; i < N; i++)
  a[i] = i;
```

2.
```
for(i = 0; i < N; i++)
  for(j = 0; j < N; j++)
    b[i][j] = function_call();
```

3.
```
for(i = 0; i < N; i++)
  b[i][k] = b[a[i]][k];
```

4.
```
for(i = 1; i < N; i++)
  a[i] = a[i-1] + b[i][k];
```

5.
```
for(i = 0; i < N-abs(k); i++)
  b[i][k] = b[i+abs(k)][k] – a[i];
```

6.
```
for(i = N/4; i < N/2-k; i++)
  a[i] = a[i+k];
```

7.
```
for(i = 0; i < N; i++)
    a[4*i] = a[2*i-1]
```

8.
```
for(j = 0; j < N; j++)
  b[i][j] = b[i-1][j];
```

9.
```
for(i = 0; i < N-1; i+=2)
  for(j = i; j < N; j++)
    b[i][j] = b[i+1][j-1];
```

10.
```
for(i = 0; i < N-1; i++) {
  b[i+1][k] = a[i+1];
  a[i] = b[i][k];
}
```

11.
```
for(i = N/2; i < N; i++) {
  a[i-N/2] = a[i];
  sum += a[i];
}
```

12.
```
for(i = 0; i < N-1; i+=2)
  a[i] = a[i+1];
```

13.
```
for(i = 2; i < N; i++)
  for(j = 0; j < M; j++) {
    a[i] = b[j+3][i-2] - 99;
    b[j+1][i] = x + y * i;
}
```

14.
```
for(i = 2; i < N; i++)
  for(j = 1; j < M; j++)
    for(k = 0; k < P; k++)
      a[i-2][2*j][3*k+3] = a[i][2*j-2][3*k+15] + 4;
```

15.
```
for(k = 1; k < 100; k++)
  for(j = 1; j < 100; j++) {
    b[1][j][k] = a[1][j-1][k];
    for(i = 1; i < 100; i++)
      a[i+1][j][k] = b[i][100-j][k];
  }
```

1. Classify the dependences for accessing the array elements;
2. Indicate the distance and the direction vectors for accessing the array elements;
3. Parallelize the codes using the Intel or the PGI compilers. Restructure the code and use compiler-specific parallelisation pragmas (no OpenMP ones) to help the compiler, if necessary;
4. Execute the codes on one parallel machine using 1, 2, 4, and 8 threads and measure the speedup and efficiency.