Aufgabe 1. Lade dir Cygwin herunter und installiere das Paket für OpenMP. Kompiliere folgendes Beispielprogramm

```
#include <stdio.h>
#include <omp.h>
int main() {
  double a [10000];
  #pragma omp parallel
  {
    int i;
    #pragma omp for
    for (i=0; i< 10000; i++)
      a[i] = i*i/2.0;
    #pragma omp master
    {
      printf("%f\n", a[3432]);
    }
  }
  return 0;
}
```

Aufgabe 2. Schreibe eine parallelisierte Funktion zur Multiplikation von Matrizen.

Aufgabe 3. Implementiere den Strassen-Algorithmus zur schnellen Matrixmultiplikation, natürlich parallelisiert.

Für $C = A \cdot B$ mit $A, B, C \in \mathbb{R}^{2^n \times 2^n}$ sei folgende Partitionierung gegeben:

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \qquad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \qquad C = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix}$$

wobei alle Untermatrizen gleich groß sind. Definiere 7 neue Matrizen:

$$\begin{aligned} M_1 &:= (A_{1,1} + A_{2,2}) \cdot (B_{1,1} + B_{2,2}) \\ M_2 &:= (A_{2,1} + A_{2,2}) \cdot B_{1,1} \\ M_3 &:= A_{1,1} \cdot (B_{1,2} - B_{2,2}) \\ M_4 &:= A_{2,2} \cdot (B_{2,1} - B_{1,1}) \\ M_5 &:= (A_{1,1} + A_{1,2}) \cdot B_{2,2} \\ M_6 &:= (A_{2,1} - A_{1,1}) \cdot (B_{1,1} + B_{1,2}) \\ M_7 &:= (A_{1,2} - A_{2,2}) \cdot (B_{2,1} + B_{2,2}) \end{aligned}$$

daraus ergibt sich dann für die Lösung C:

$$\begin{aligned} C_{1,1} &= M_1 + M_4 - M_5 + M_7 \\ C_{1,2} &= M_3 + M_5 \\ C_{2,1} &= M_2 + M_4 \\ C_{2,2} &= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

Dabei soll zur Berechnung der M_i natürlich wieder der gleiche Algorithmus zur Matrixmultiplikation verwendet werden. Man spart insgesamt eine Matrixmultiplikation und verringert so den Aufwand von $O(n^3) = O(n^{\log_2(8)})$ auf $O(n^{\log_2(7)})$ also ungefähr $O(n^{2,807})$.