Parallel Computing Exercise 1

1 Task 1 - Run-Time influencing factors

- Algorithmic complexity of the program (notation: $O(g(n))$, or $\Theta(g(n))$)(high):
  This is a very high factor for program runtime as this is an estimation of the 'number of steps' that the algorithm has to make in order to produce the output. The importance of algorithmic complexity can be seen by the fact that one of the famous 'Millenium Problems' is indeed the question if $P = NP$.

- Input size for the algorithm (the n in $O(g(n))$)(high):
  Also a very important factor. For example if you want to solve a partial differential equation on a regular grid it will cost more time and memory if you have 10, 10000 or 10^9 grid points.

- Underlying hardware (meaning number of transistors of CPU/GPU, clock rates, bus clock, etc.)(normal):
  Better hardware has made previously infeasible problems solvable for even moderately priced consumer hardware in recent years. Thus the demand for better runtime can be approached by consecutively better hardware, but a less complex algorithm or a different approach to the problem (see points 1 and 2) can have considerably better effects. There are even problems (like integer factorization for moderately large integers of say 500 decimal digits) for which there is no computer and known algorithm in existence, which can solve this problem in practical time. So better hardware can simplify many practical problems but some problems are so inherently difficult that probably no computer will calculate a solution to those in a practical span of time.

- Program language used (low):
  The use of a specific program language can have (even noticeable) impact on the run time of a program. Overhead like garbage collection or interpreting the code can slow down the execution of the program considerably, but neither does change the algorithmic complexity or the underlying hardware, so a change from a 'slow' to a 'fast' language doesn't have as much effect as changing one of the above points.