worksheet 4

by **Konstantin Devyatov**

Part I

Problem 1.1

See folder **../1**

Part II

Problem 1.2

See folder **../2**

Part III

Problem 1.3

Measurement

number of threads / implementation	sequential	parallel (creation of 10K threads) (in s)	parallel (using one thread for all iterations on a chunk)
1	0.313,921	2.586,987	0.326,875
2	0.308,483	2.422,732	0.167,585
5	0.309,331	6.429,677	0.169,293
10	0.308,600	16.442,160	0.169,071
15	0.308,405	25.130,958	0.176,048
20	0.308,325	33.661,502	0.172,082
25	0.309,178	42.241,863	0.166,965
30	0.309,337	51.041,486	0.169,510

All tests have been performed on linux18 lab machine: centOS 6.3 x64, 3.7 GiB of RAM, dual-core 3 GHz CPU.

Observations

From the measurements i was able to observe that the technology behind pthread_create does not reap time-efficiency benefits on its own. Which means that programming ingenuity is still highly valued, especially so with the market approachign limitations of the processor clock speed.

Specifically, i saw that 2 threads provide the most time-efficient solution to this trivial problem. That is because the of two reasons:

- with the number of threads resource overhead increases and when n>2, the overhead starts to overtake the benefit gained by parallelisation
- once a thread is finished it need to wait for other threads, in arbitrarily selected order, to finish, before it could terminate; this also creates workload bubbles, decreasing overall time-efficiency