1 Question 1

1.1 Task a)

The program was implemented as instructed and is attached to this submission.

1.2 Task b)

The results of running the compiled program on Euler and my own machine (a 2018 Macbook Pro) are plotted in figures 1 and 2. at the marks seen in the random task, but they are much less pronounced: While we're able to read from cache lines, reading the increasingly large arrays makes memory access a bottle neck.

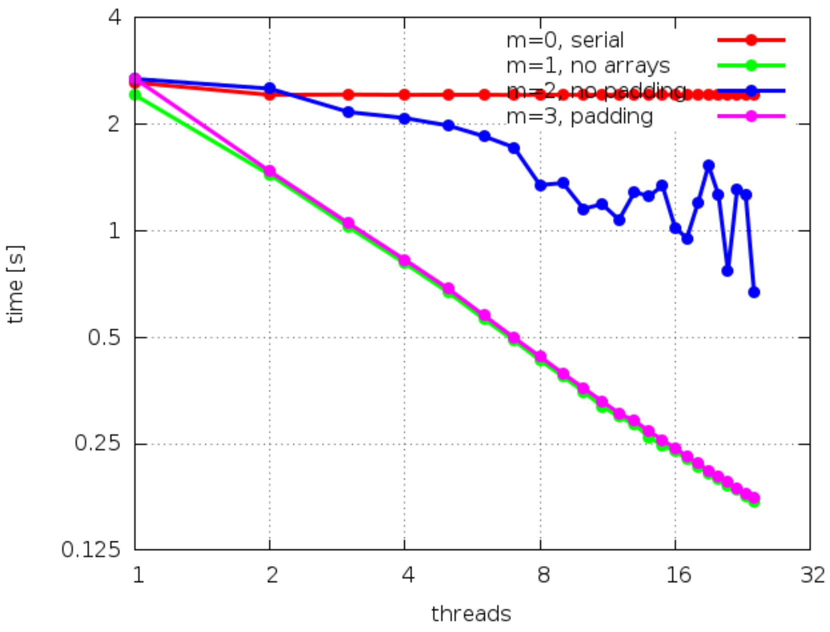

Figure 1: blub Euler compute cluster.

2 Question 2

There is a problem with the global variable `pos` in the `if` statement starting on line 13: In order to write to the array `good_members`, the value of `pos` must be read in line 14. This read is not protected from race conditions with the atomic write on line 17. As `#pragma omp atomic` cannot be extended to several statements, a possible (but still ugly) solution would be to replace the `for` loop block from lines 12 to 19 with snippet 1.

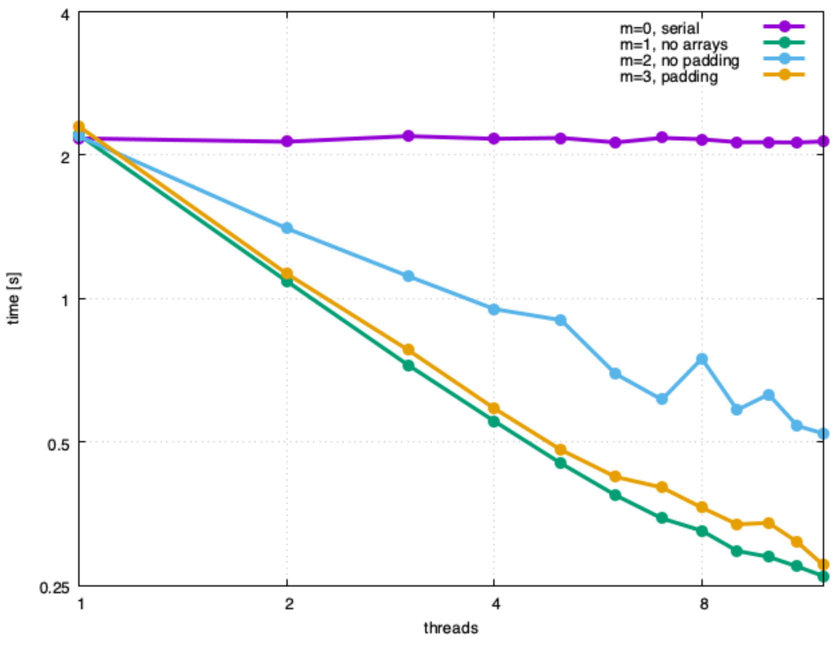

Figure 2: blub 2018 Macbook Pro cluster.

Listing 1: 'Blub'

```
#pragma omp parallel for
for (int i= 0; i < N; i++)
{
    if (is_good(i)) // No change until here
    {
        int thread_pos{0}; // Define thread specific position variable
        #pragma omp critical // This block replaces the #pragma omp atomic section
        {
            thread_pos= pos; // Protected read
            pos++; // Protected write
        }
        good_members[thread_pos]= i; // Use thread specific variable
    }
}
```