Assignment 2

Eric Abraham (1408828)
e.abraham@student.tue.nl

Matthaios Biskas (1430033)
m.biskas@student.tue.nl

December 22, 2021

1 Threads and synchronization

All the threads are used in detached state, as this provides a cleaner way of dealing with the semaphore. When a thread finishes executing it's task, it increases the semaphore and terminates itself. This simplified the logic for us as there was no need to keep track of when a thread needs to be joined.
A semaphore is used in order to ensure that at most 10 threads are active at one given time. The threads are stored in an array, and based on the current value of the semaphore, a thread is created given a pointer to an 'available' address space in the array.
The routine is passed an address in memory that corresponds to the number of a piece. Inside the routine a loop is used in order to iterate over all the multiples of the given piece. As the bit data represents a shared resource, a mutex is needed in order to ensure mutual exclusion between thread operations on the bits. After the bits of the multiples are flipped, the memory address is released, the semaphore increment operation is used and the thread exits.

2 Helper code

Firstly, we use a function that sets all the bits to 1. The size of the buffer is computed in a dynamic manner, in order to improve usability.
Another function simply goes over all the pieces and prints a piece if the bit is set.
The function for flipping the bit contains a ternary operator that simply sets/clears a bit at a specified index, depending whether the bit was set or not.