LAB EXAM
Q1

by: Devraj, ES22BTECH11011

Design of the program

In this program we use multithreading to find perfect numbers from 1 to N.

Two files are opened for input and output respectively. We take the variables N and K from the input file.
Then we define an array of threads of size K. Each thread executes a runner funtion.
A structure named arguments is created to store the arguments to be passed into the runner function.

To do this an array of arguments is defined with each index corresponding to a thread.
The arguments are:
1. start: the lowest number in the set corresponding to a thread
2. max: the maximum allowable number that can be checked
3. jump: the value by which a number differs from the previous number
4. thread_no: the index of the thread to which this argument will be passed

The set of numbers alloted to a thread is determined as follows:

thread number i receives the numbers: {i, i+K, i+2K, ..., i+[N/K – 1]*K};

i.e., the entries in the set of numbers for each thread are separated by atleast K.
As we can see in the output file, the perfect numbers are not uniformly distributed in 1 to N, they actually appear quite randomly in any range of numbers.

Hence, dividing these N numbers into chunks like
{1,2,3,...,N/K}, {N/K+1,..., 2N/K} and so on
will not be efficient as there might be no perfect numbers in one of these sets and a few in another.

Outputs

Time vs Size (N)

N	1000	10000	100000	1000000
time(ms)	0.35	7.19	590.73	60193.2

As we can see, the time of execution increases as the size of the range N increases. This behavior is expected as each thread now has to do more work.

Time vs No. of Threads (K)

K	1	2	4	8	16
time(ms)	3383.92	1650.12	865.462	575.809	434.387

As we can see the time for execution decreases as we increase the number of threads K. This is because now the work is divided between multiple threads which can work in parallel, thus reducing time.