Programming Assignment-1 Report
CS3510: Operating Systems-1
Computation of Perfect Numbers

Name: Varunaditya Singhal
Roll Number: MA20BTECH11021

Question: In this assignment, I will develop a multi-threading program that checks all the perfect numbers, less than n, for a given positive integer N.

Given: The Perfect Number N is stated as a number equal to the sum of all its factors except that number. For example, if n = 6, the factors are: 1, 2, 3, 6.
So Sum except 6 = 1+2+3.

I had to write a C program to implement this using multi-threading, ensuring that each thread runs a unique set of numbers < n. The results of each thread are displayed as “OutFile_i” for the ith thread.

One OutMain file is generated, which contains the data from each thread and displays the total Perfect numbers for the given input.

Low-Level Design: While compiling the code, the program takes the input from the “input.txt” file, which contains two positive integers N and K. K is the number of threads to be generated. If any of the integers is negative, the program displays an error message and exiting code, so that the user can change the input file data and re-run the code.

After taking the inputs, the start and count values are declared. If k doesn’t divide n, then the ceil of (n/k) is taken. For inputs like 30 17, the first 15 threads will take count as 2 and 2 values will be checked with each thread. The last 2 threads will be empty.

Then memory is allocated to these thread parameters, and thread_runner is called to run an individual thread with the algorithm of Perfect numbers with the numbers allocated to it.
The thread_runner deposit the result of each value on an OutFile and adds the Perfect Numbers (if found) to storage. Each thread ends by freeing the memory of the File pointer. When all the threads are run and the OutFile of all threads is created, the program uses the storage of each thread and adds those values to an OutMain file.

The process ends with freeing the memory of the pointers to the files and storage parameters.

Sample Input in input file:
10 10

Sample Output: (on running ./Assgn2Src-MA20BTECH11021)

Initially, the “input.txt” file contains 10 10 as the data.

A total number of threads created in this case is 10 and each thread contains ceil(n/k) i.e 1 value.

- **In OutFile_1.txt :**
  - 1 : Not a Perfect Number.
- **In OutFile_2.txt :**
  - 2 : Not a Perfect Number.
- **In OutFile_3.txt :**
  - 3 : Not a Perfect Number.
- **In OutFile_4.txt :**
  - 4 : Not a Perfect Number.
- **In OutFile_5.txt :**
  - 5 : Not a Perfect Number.
- **In OutFile_6.txt :**
  - 6 : Is a Perfect Number.
- **In OutFile_7.txt :**
  - 7 : Not a Perfect Number.
- **In OutFile_8.txt :**
  - 8 : Not a Perfect Number.
- **In OutFile_9.txt :**
  - 9 : Not a Perfect Number.
- **In OutFile_10.txt :**
  - 10 : Not a Perfect Number.
- **In OutMain.txt :**
  - Thread1 :
  - Thread2 :
  - Thread3 :
  - Thread4 :
  - Thread5 :
  - Thread6 : 6
  - Thread7 :
  - Thread8 :
  - Thread9 :
  - Thread10 :

Analysis of Sample Output: The value 10 is stored in base_number and another value 10 is stored as the number of processes. Now, it checks the count of values to be added to each thread which comes out to be ceil(10/10) =1. Thus each thread has 1 value given to it. Demonstrating what the algorithm does for each value inside each thread - Let's say the value is 6, the code adds the number i < 6 , if it divides 6 , if the final sum of i's ==6. We say that 6 is a perfect number. Proceeding this with every number on each thread, the program creates Output files to add data in them and finally an OutMain file to share the perfect numbers found by each of the threads. Finally ending the program after freeing the non-useful memory pointers.