OS1 – CS3510

Report: Programming Assignment 2 - Finding Perfect numbers using threads.

Aditya Sridhar

ES21BTECH11004

Problem Statement:

Our problem is to find all perfect numbers between 1 and the number ”n”. This is done by partitioning the work amongst “k” threads created by the main thread, such that no two threads work on the same number (by work, we mean finding whether the numbers assigned to that thread are perfect or not). “k” and “n” are input variables.

Brief Summary:

The k threads were allocated the numbers from 1 to n as follows:

Thread 1 – $\{1, k + 1, 2k + 1, 3k + 1, \ldots, k*\lfloor n/k \rfloor + 1\}$

Thread 2 – $\{2, k + 2, 2k + 2, 3k + 2, \ldots, k*\lfloor n/k \rfloor + 2\}$

Thread 3 – $\{3, k + 3, 2k + 3, 3k + 3, \ldots, k*\lfloor n/k \rfloor + 3\}$, etc

This would be the allocation, provided none of these numbers exceed n. This makes sure that each thread, to some extent, while not doing the work of any other thread, does around the equal amount of work as the other threads due to near-equal distribution.

Detailed Description of Algorithm:

- We first open the input file that contains the values of n and k. We take this file as a command line argument.
- In the occurrence that k is greater than n, we require only n threads (each thread will take care of one number). In this case, all threads between n + 1 and k are not required. This is printed in the OutMain.txt
- We now create a pointer-“parameters”, which stores the values of n, k, and i (i is the counter variable for the threads). Since pthread_create takes a (void *) parameter, this (int *) pointer is typecasted to (void *) and then passed to the function threadAlloc_perfCheck.

- Function **threadAlloc_perfCheck**:

  - This function does the following – using the passed parameters n, k and i (this is unique for each thread), the function computes all numbers it has to check for whether it is perfect or not. **(Math. 1)**
  - The thread checks whether the number is perfect or not using the function ifPerf. (**Math. 2)**
  - These perfect numbers are stored in an array isPerf. This array has a maximum length of $\lfloor n/k \rfloor$. The number of elements in the array is also stored in an integer variable – numOfPerf.
  - All numbers assigned to Thread$_i$ are stored in a file OutFile$_i$ – This file will store whether each number is a perfect number or not.
  - To the main file OutMain.txt, we append all the newly found perfect numbers by that thread. After the completion of all threads, OutMain.txt will contain each thread, followed by all the perfect numbers they found among the numbers they were assigned.

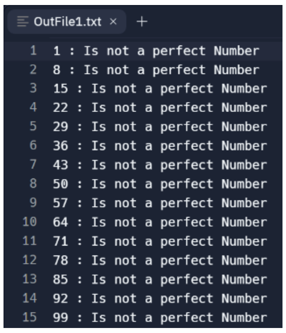
Math.1:
The algorithm for allocating the numbers amongst the threads is pretty straight-forward. **Thread$_i$ will have an "adder"** assigned to it. This will be equal to i (i ranging from 1 to k). To get the numbers for each thread described in the **Brief Summary**, we go in a loop, taking j as the looping variable.
We can see that **the last term in all the numbers assigned to Thread$_i$ is the adder**. Hence, the numbers will be **j*k + adder**, j ranging from 0 to $\lfloor n/k \rfloor$, including 0 and $\lfloor n/k \rfloor$**.**
In every iteration, we make sure that the number it is checking is less than n. If not, it goes to the next number.
This way, we **near-equally distribute** all numbers 1 to n amongst k threads.

Math. 2:
The algorithm to check whether a given number is a perfect number is fairly simple. We find out all factors of the number and add them to the variable sumOfFact. In the end, if the sum of its factors (excluding itself) is equal to the number itself, the function returns 1; if not, it returns 0.

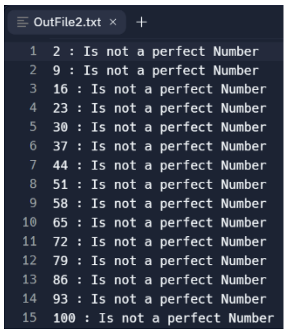
Analysis of Output:

1. **Input: n = 100, k = 7**

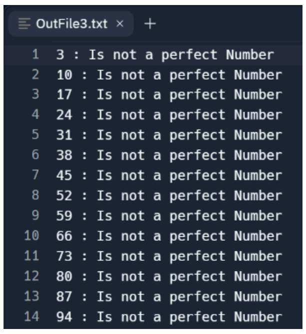
   Output:



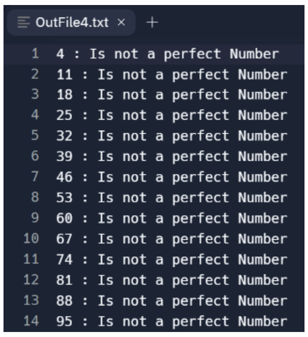
```
1 : Is not a perfect Number
8 : Is not a perfect Number
15 : Is not a perfect Number
22 : Is not a perfect Number
29 : Is not a perfect Number
36 : Is not a perfect Number
43 : Is not a perfect Number
50 : Is not a perfect Number
57 : Is not a perfect Number
64 : Is not a perfect Number
71 : Is not a perfect Number
78 : Is not a perfect Number
85 : Is not a perfect Number
92 : Is not a perfect Number
99 : Is not a perfect Number
```



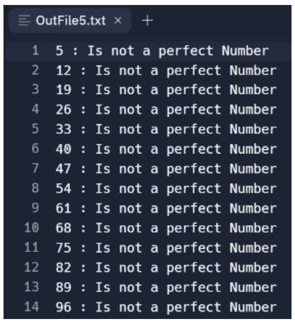
```
2 : Is not a perfect Number
9 : Is not a perfect Number
16 : Is not a perfect Number
23 : Is not a perfect Number
30 : Is not a perfect Number
37 : Is not a perfect Number
44 : Is not a perfect Number
51 : Is not a perfect Number
58 : Is not a perfect Number
65 : Is not a perfect Number
72 : Is not a perfect Number
79 : Is not a perfect Number
86 : Is not a perfect Number
93 : Is not a perfect Number
100 : Is not a perfect Number
```



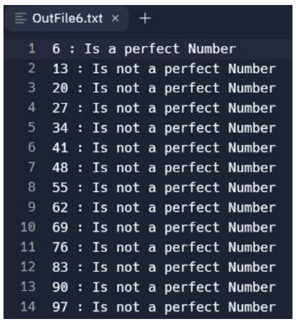
```
3 : Is not a perfect Number
10 : Is not a perfect Number
17 : Is not a perfect Number
24 : Is not a perfect Number
31 : Is not a perfect Number
38 : Is not a perfect Number
45 : Is not a perfect Number
52 : Is not a perfect Number
59 : Is not a perfect Number
66 : Is not a perfect Number
73 : Is not a perfect Number
80 : Is not a perfect Number
87 : Is not a perfect Number
94 : Is not a perfect Number
```



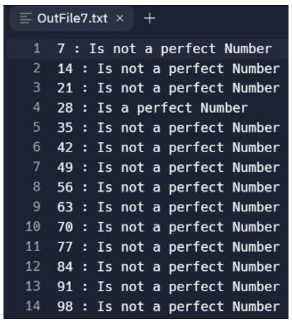
```
4 : Is not a perfect Number
11 : Is not a perfect Number
18 : Is not a perfect Number
25 : Is not a perfect Number
32 : Is not a perfect Number
39 : Is not a perfect Number
46 : Is not a perfect Number
53 : Is not a perfect Number
60 : Is not a perfect Number
67 : Is not a perfect Number
74 : Is not a perfect Number
81 : Is not a perfect Number
88 : Is not a perfect Number
95 : Is not a perfect Number
```



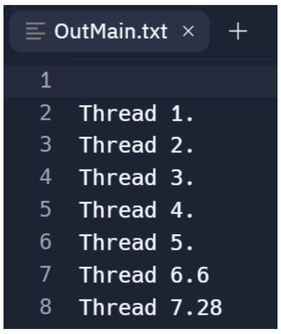
```
5 : Is not a perfect Number
12 : Is not a perfect Number
19 : Is not a perfect Number
26 : Is not a perfect Number
33 : Is not a perfect Number
40 : Is not a perfect Number
47 : Is not a perfect Number
54 : Is not a perfect Number
61 : Is not a perfect Number
68 : Is not a perfect Number
75 : Is not a perfect Number
82 : Is not a perfect Number
89 : Is not a perfect Number
96 : Is not a perfect Number
```



```
6 : Is a perfect Number
13 : Is not a perfect Number
20 : Is not a perfect Number
27 : Is not a perfect Number
34 : Is not a perfect Number
41 : Is not a perfect Number
48 : Is not a perfect Number
55 : Is not a perfect Number
62 : Is not a perfect Number
69 : Is not a perfect Number
76 : Is not a perfect Number
83 : Is not a perfect Number
90 : Is not a perfect Number
97 : Is not a perfect Number
```



```
7 : Is not a perfect Number
14 : Is not a perfect Number
21 : Is not a perfect Number
28 : Is a perfect Number
35 : Is not a perfect Number
42 : Is not a perfect Number
49 : Is not a perfect Number
56 : Is not a perfect Number
63 : Is not a perfect Number
70 : Is not a perfect Number
77 : Is not a perfect Number
84 : Is not a perfect Number
91 : Is not a perfect Number
98 : Is not a perfect Number
```



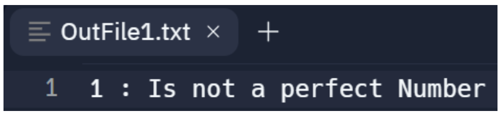
```

Thread 1.
Thread 2.
Thread 3.
Thread 4.
Thread 5.
Thread 6.6
Thread 7.28
```

Analysis:

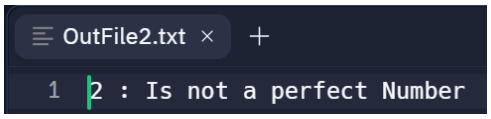
- We know that the only perfect numbers between 1 and 100 are 6 and 28. Hence, that computation has been done properly (Math. 2).
- We see that Threads 1 and 2 have computed 99 and 100. However, the subsequent threads have **NOT computed 101, 102, and so on**. Hence, that condition has been applied properly.
- The allocation has also been done properly. According to our algorithm,
  Thread$_6$: {**6**, 13, 20, 27, …}
  Thread$_7$: {7, 14, 21, **28**, 35, …}
- As expected, Threads 6 and 7 have detected 6 and 28 to be perfect numbers, respectively.

2. Input: n = 7, k = 9

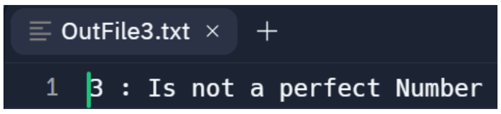
Output:


OutFile1.txt

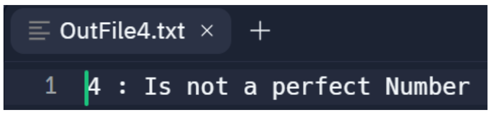
```
1 : Is not a perfect Number
```


OutFile2.txt

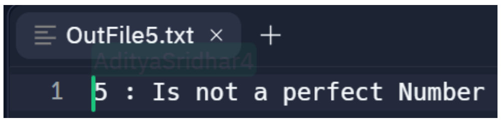
```
2 : Is not a perfect Number
```


OutFile3.txt

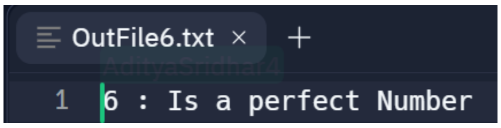
```
3 : Is not a perfect Number
```


OutFile4.txt

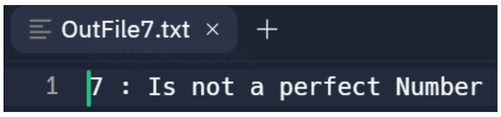
```
4 : Is not a perfect Number
```


OutFile5.txt

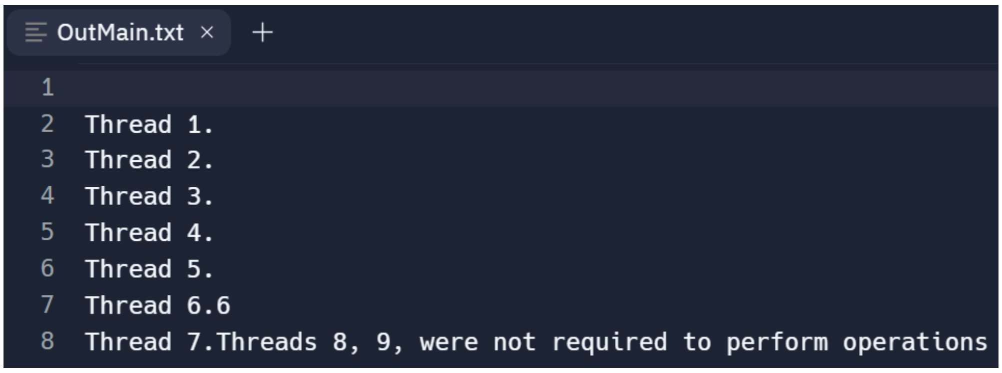
```
5 : Is not a perfect Number
```


OutFile6.txt

```
6 : Is a perfect Number
```


OutFile7.txt

```
7 : Is not a perfect Number
```


OutMain.txt

```

Thread 1.
Thread 2.
Thread 3.
Thread 4.
Thread 5.
Thread 6.6
Thread 7.Threads 8, 9, were not required to perform operations
```

Analysis:

- The only perfect number between 1 and 7 is 6. This is detected by Thread$_6$.
- The allocation has been done properly. According to our algorithm,
  Thread$_1$: {1}
  Thread$_2$: {2}
  Thread$_6$: {**6**}
- Also, as per our code, threads 7, 8, and 9 are **not required and are hence not used**. This is also printed in the OutMain.txt file.