ASSIGNMENT - 1

DESCRIPTION:

Low-Level Design of the Program:

My C program is to find and mark tetrahedral numbers within a specified range using parallel processing with multiple child processes. This program takes input values N and K from an input file, where N is the upper limit of the range of numbers to check and K is the number of child processes to be created. Then it uses shared memory to communicate results between the main process and the child processes.

Breakdown of the low-level design:

1. **Input Reading:**
    - The program reads N and K from the "input.txt" file.
    - It validates the input values, ensuring that N and K are positive integers within reasonable bounds.
2. **Shared Memory:**
    - The program creates a shared memory segment using `shm_open` to facilitate communication between the main process and child processes.
    - The shared memory is allocated to hold an array of integers, each representing whether a corresponding number is a tetrahedral number or not.
3. **Forking Child Processes:**
    - The program forks K child processes using a loop.

    - Each child process is responsible for a specific range of numbers within the overall range from 1 to N.
    - The child processes calculate whether each number in their range is a tetrahedral number or not and store the result in a local array.
4. **Child Process Function:**
    - The `childProcess` function takes the start and end values of the range assigned to the child process and calculates tetrahedral numbers in that range.
    - The results are stored both in a local array and the shared memory.
5. **Output Handling:**
    - After all child processes have completed, the main process waits for them using the `wait` system call.
    - It then consolidates the results from shared memory and writes them to an output file.
    - The program creates individual log files for each child process, containing details about whether each number in its range is a tetrahedral number or not.
6. **Main Log File:**
    - The program creates a main log file ("OutMain.txt") containing a consolidated list of tetrahedral numbers across all child processes.
7. **Memory Management:**
    - The program detaches and removes the shared memory segment.

Complications and Analysis:

1. **Concurrency Issues:**

- The program utilizes parallel processing to improve efficiency by distributing the workload among multiple child processes. However, concurrent access to shared resources, such as the shared memory, requires careful synchronization to avoid race conditions. The program appears to handle this through separate ranges assigned to each child process.

2. **Memory Management:**
   - Proper memory management is crucial in a program using shared memory and forked processes. The program allocates and frees memory appropriately in both the main and child processes to prevent memory leaks.
3. **Output Analysis:**
   - The program generates individual log files for each child process, facilitating debugging and analysis. Additionally, a main log file consolidates results, providing a comprehensive overview of tetrahedral numbers in the specified range.
4. **Code Readability:**
   - While the code is functional, readability could be improved. Meaningful comments, clear variable names, and structured code can enhance maintainability and make it easier for others to understand the implementation.
5. **Error Handling:**
   - The program includes error handling for file operations and shared memory creation. However, comprehensive error handling throughout the program could further enhance its robustness.

Summary :

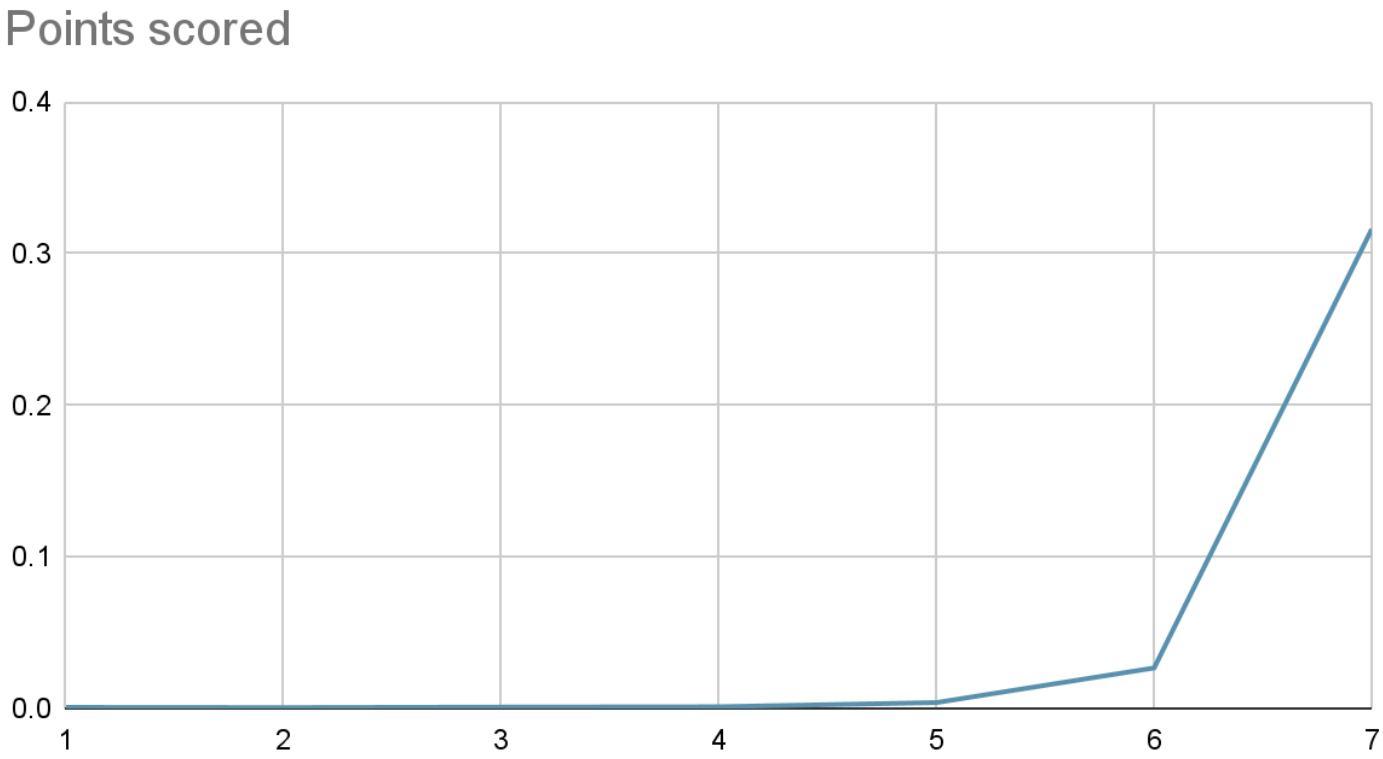
My program effectively utilizes parallel processing and shared memory to identify and log tetrahedral numbers within a specified range. Careful consideration of concurrency and memory management contributes to the program's correctness.

GRAPHS :

1. Time vs Size, N:

In this graph, the y-axis will have the time taken by algorithm. The x-axis will be the values of n varying from 1 to 7 in increments of 1. Note that N=2^3n . Have K fixed to be 8 for all these experiments.

n	K	N	Time taken by algorithm in seconds
1	8	8	0.003204
2	8	64	0.002796
3	8	512	0.002989
4	8	4096	0.001244
5	8	32768	0.003998
6	8	262144	0.026820
7	8	2097152	0.316095



2. Time vs Number of Processes, K:

In this graph like the previous graph, the y-axis will have the time taken by your algorithm. The x-axis will be the values of K, number of processes varying from 1 to 16 (in powers of 2 i.e, 1,2,4,8,16). Have to be N fixed at 1000000 (10 lakhs) for all these experiments.

K	N	Time taken by algorithm in sec
1	1000000	0.136692

2	1000000	0.097682
4	1000000	0.078121
8	1000000	0.062073
16	1000000	0.051676