Operating System-1
ASSIGNMENT-1 REPORT
SURBHI (CS22BTECH11057)

1.Introduction:-
It is multi-process program designed to identify the tetrahedral number within a given range.Tetrahedral numbers are the sequence of figurate numbers that represent the number of dots in a three-dimensional triangular pyramid. The program employed forked processes to parallelize the computation, enhanching performance by distributing the workload across multiple cores. The report includes the analysis of program's performance through two specific graphs:- Time vs Size (N) and Time vs Number of Processes (K).

2. Program Overview:-
The program is written in C Program, begins with reading input parameters (N) and (K) from an input file named "input.txt". (N) represent the range of numbers and (K) represent number of child processes to be spawned. "ChildProcess" functions checks whether each number in its assigned range is a tetrahedral number, if it is a tetrahedral number it is recorded in a log file associated with respective child process. After child process complete their execution, the main process waits for them to finish using "wait()". It then consolidates the results from individual log files into a main output file named "OutMain.txt". It provides a summary of identified tetrahedral numbers, including the process number responsibile for each finding. I used "ftruncate" is used to set the size of the shared memory region created with "shm_open" The purpose of ftruncate is to define the size of the shared memory segment. "ftruncate" sets the size of the shared memory object to a specified length. In this case, the size is set to "(sizeof(int) * N *2)" which is the total size needed for storing the array of numbers and the array of results. The "ftruncate" function helps ensure that the shared memory segment is large enough to accommodate the data. I used "mmap" for sharing memory object into the address space of the calling process and it returns a pointer to the mapped area. The "PROT_READ | PROT_WRITE" falgs specify that the mapped region should be readable and writeable.

3.Low-Level Design:-
(a) **"is_tetraheadral(int num)"** function systematically determines whether a given number qualifies as a tetrahedral number, in a loop. It iterates through tetrahedral candidates until a match is identified or until the number surpasses the specified target.

(b) **"find_tetrahedral(int start, int end, int process_id)"** function systematically explores the range[start, end] to identify tetrahedral numbers.

(c) **“main”** function reads N and K from “input.txt” file, and program calculates the appropriate range for each process based on N and K.
K processes are spawned through forking. Each child process invokes “find_tetrahedral()” function to analyze its assigned numneric range.

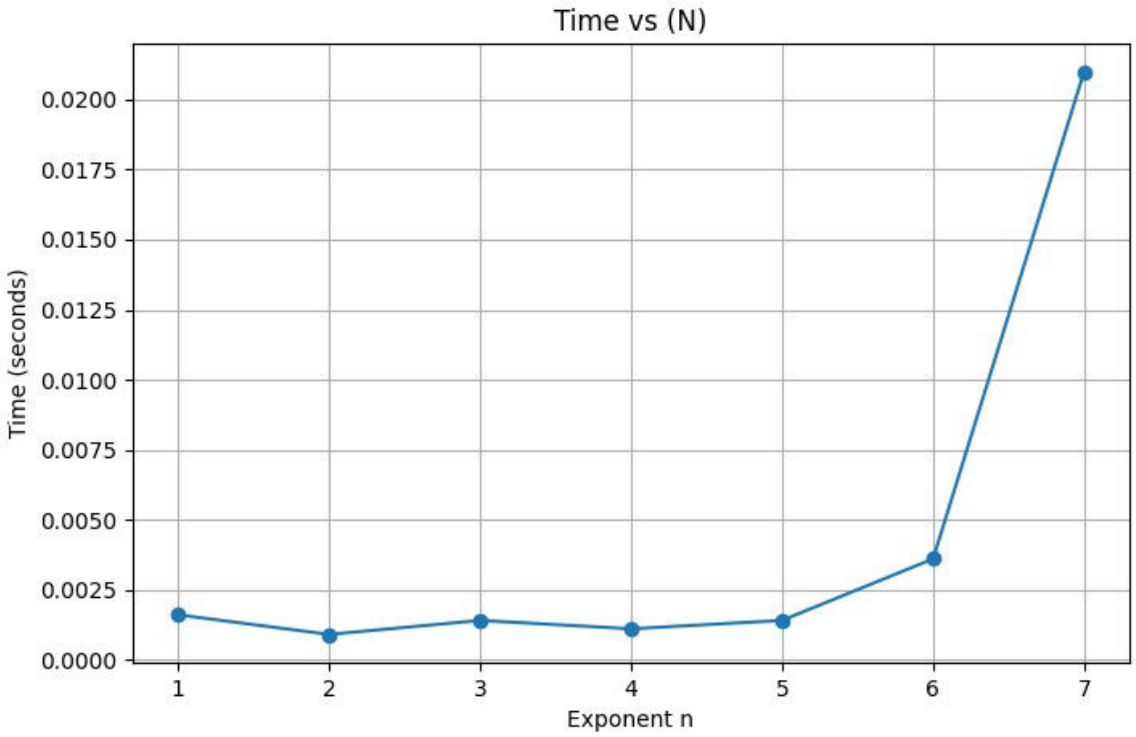
4.Error Handling:-

(a) **Shared Memory Allocation:** It checks if there is an error during an allocation of shared memory using “shmget”. If an error occurs, it prints an error message using “perror” and exits the program with a failure start “EXIT_FAILURE”. This ensures that the program does not proceed with faulty shared memory allocation.

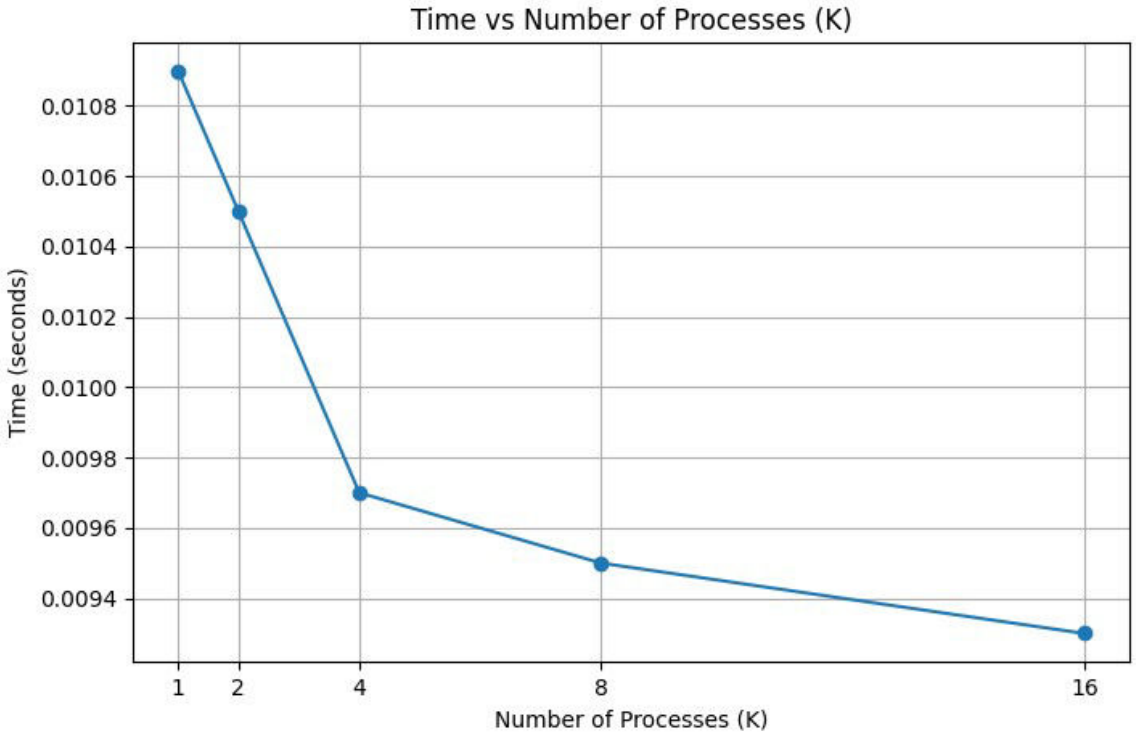
(b) **File Operation:-** It checks for errors during file operation. If there is an issue opening “input.txt” file, appropriate error messages are displayed, and the program exits with a failure status. This prevent from proceeding with invalid or missing input data.
(c) **Forking Processes:-** It checks for error during the forking of child processes. If there “fork” system call fails, an error message is displayed using “perror” and program cleans up any shared memory and exits with failure status.

5. Performance Analysis:-

(a) TIME vs SIZE (N)



(b) TIME vs NUMBER OF PROCESSES (K)



6. Conclusion:-

This program efficiently utilizes parallel processing to find tetrahedral numbers within a given range. The provided graphs offer insights into the program's performance under varying input conditions. Further optimizations could focus on reducing memory usage and enhancing synchronization mechanisms for improved scalability. The report provides a comprehensive overview of the program's design, challenges faced during development, and a detailed analysis of its performance.