IPC Tasks

1. Write a program that takes two command-line arguments: an input file and an output file. The program should copy the contents of the input file to the output file using read() and write() system calls. Make sure to handle errors if the files cannot be opened.
2. Write a C program where a child process sends a message “Hello from child!” to the parent using a single pipe. The parent process should read the message sent by the child and print it to the console. Ensure that the child writes to the pipe, and the parent properly reads from it, handling any necessary closing of unused pipe ends.
3. Develop a program in which the parent process sends an integer number to the child process through a pipe. The child should read the number, compute its square, and send the squared value back to the parent. The parent process should then read and print the squared result. Make sure to handle proper synchronization so that the processes communicate efficiently.
4. Extend the previous problem to involve two child processes. The parent process should send a number to both children. The first child computes the square of the number and sends it back, while the second child computes the cube of the number and returns the result to the parent. The parent should collect both results and print them. Use two separate pipes for communication between the parent and each child
5. Implement a parallel sum computation program using four child processes. The parent process initializes a large array of numbers and divides it into four equal parts, assigning each part to a different child process. Each child computes the sum of its assigned portion and sends the partial sum back to the parent through a pipe. Once the parent receives all four partial sums, it computes the total sum and prints the final result. Ensure efficient handling of the array partitioning and communication between processes.
6. As a challenge, write a program where a parent process distributes the computation of a large factorial to four child processes. The parent sends the value of `N` and divides the factorial computation among the four children, where each child calculates a portion of the factorial. The first child computes the product of numbers from `1` to `N/4`, the second from `(N/4)+1` to `N/2`, the third from `(N/2)+1` to `(3N/4)`, and the fourth from `(3N/4)+1` to `N`. Each child then sends its computed partial factorial back to the parent, which multiplies all the received results to obtain the final factorial value of `N`. The program should ensure efficient parallel execution and correct handling of large numbers.

Instructions for Submission

- Ensure your code is well-documented with comments.
  Implement proper error handling (e.g., file not found, read/write failure, pipe issues).
- Test your program with **various inputs** before submission.
- Submit your **source code along with screenshots of successful runs**.