1. How many threads are you going to use? Specify the task that you intend each thread to perform.
I'm going to use 3 threads in total, one per flow.

2. So the threads work independently? Or, is there an overall "controller" thread?
This, they work independently. No controller

3. How many mutexes are you going to use? Specify the operation that each mutex will guard.
I'm going to use one mutex, to be performed as a lock to the critical section.

4. Will the main thread be idle? If not, what will it be doing?
Yes, no main thread will be there.

5. How are you going to represent flows? What type of data structure will you use?
I'll define a flow data structure. There will be 4 parameters in each flow, including id(int), arrival time(float), transmission time(float), and priority(int).

6. How are you going to ensure that data structures in your program will not be modified concurrently?
Each time when modify, use mutex to lock it, and after the process is done, unlock it.

7. How many convars are you going to use? For each convar:
there will be one convar.

(a) Describe the condition that the convar will represent.
When the pipe is empty, the convar will signal, otherwise the convar will wait.

(b) Which mutex is associated with the convar? Why?
Yes,it is. Because as a shared variable by multiple threads, a convar is used along with a mutex.

(c) What operation should be performed once pthread_cond_wait() has been unblocked and re-acquired the mutex?
Signal operation
pthread_cond_signal(myConvar)
8. In 25 lines or less, briefly sketch the overall algorithm you will use. You may use sentences such as: If a flow finishes transmission, release trans mutex.

First read the text file, and the int at the first line is the number of flows.
Then after that read the next three lines, put each of them into an array to store information of each flow.
Define a construct to store id, arrival time, transmission time, and priority into each flow.
After storing these data, I'm going to create three pthread for these three flows.
Then for each thread, they'll wait till their arrival time comes.
Once they are arrived, they are going to request for the pipe to run. In this case, if there's nothing in the pipe, it can run immediately, but if the pipe is occupied, they gonna wait in the queue.
There's the principle for their position in the queue list:
when multiple flows arrive at the same time and no other flow is on transmission, the arriving flow with the highest priority starts its transmission. If the have the same priority, the one that has the smallest transmission time starts its transmission. If there is still a tie, the one that appears 1st in the input file starts its transmission first.
When the flow in the pipe finished, then the first one in the queue list will start running in the pipe, so remove it from the list, and move the flowing flows up 1 position.
After all flows being executed, destroy the mutex and condition variable.