1. How many threads are you going to use? Specify the task that you intend each thread to perform.

The number of threads would be the same as number of flows that is in the input file. Each flow will create their own thread to get accurate arrival time. I am going to use usleep function to get their arrival time,transmission time and processing time. Each thread will broadcast its condition variable when it finishes its transmission time.

2. Do the threads work independently? Or, is there an overall "controller" thread?

My threads will work independently but they will share the time. when transmission pipe is unavailable and one of the threads started its transmission time, it will sort the queue according to the given rules provided in Connex website and wait for its turn. I want my threads to share the time because It would be much easier to track of the real machine time if they share the time(Time should be a global value).

3. How many mutexes are you going to use? Specify the operation that each mutex will guard.

There will be three mutexes for each thread.
First mutex will be used for the first thread to start its transmission time and let other threads wait.
Second mutex will be used for sorting, I want to sort the items one at a time to prevent my queue from storing wrong item.
Third mutex will be used to measure the processing time. Since my threads will share the time, I don't want to have more than one thread to measure its processing time at the same time.

4. Will the main thread be idle? If not, what will it be doing?

Main thread will be idle. Main thread will wait for other threads to terminate.Main thread will destroy mutex and condition variable at the end.

5. How are you going to represent flows? what type of data structure will you use?

I am going to use abstract data type to represent flows.
my structure will contain its priority,arrival time, transmission time and flow number.

6. How are you going to ensure that data structures in your program will not be modified concurrently?

There will be my mutexes to protect my data structures, and each thread will have its own data structure so it won't be modified concurrently.

7. How many convars are you going to use? For each convar:

(a) Describe the condition that the convar will represent.

There will be only one condition variable. when transmission pipe is not available, my threads will wait for transmission pipe to be available using conditional variable. As soon as the pipe is available, one of my threads will broadcast the conditional variable, so other threads can start their transmission time.

(b) Which mutex is associated with the convar? Why?

All of my mutexes that are used for sorting items is associated with condition variable because once it broadcasts, I will have sorted items in my queue array.

(c) What operation should be performed once pthread cond wait() has been unblocked and re-acquired the mutex?

if pthread cond wait() has been unblocked and re-acquired the mutex will find a thread that is at the top of the queue and start its transmission because there will be my threads waiting for cond wait() to be unblocked. if (item ->id =queue[0] ->id); I am going to use this statement to find my item.

8. In 25 lines or less, briefly sketch the overall algorithm you will use. You may use sentences such as:

struct flow contains arrival time,transmission time,priority and id;
create condition variable , mutex ,queue array
create a global time variable.

main{
for(i=0;i<the number of flows; i++)
create threads;}
thread subroutine{
start timer
sleep(arrivaltime)
sort(thread)
start timer
sleep(transmissiontime)
stop timer
inform other threads that it finished its transmission by calling cond_broadcast}
sort{
stop timer
print arrivaltime
start transmission for the thread that first arrives and return to subroutine.
lock mutex (sort items at a time)
store the other threads items in the queue array
start sorting according to the rule.
unlock mutex
while item ->id =queue[0] && the pipe is available
wait for the thread that is currently working on transmission to finish using cond_wait();
when cond_broadcast is called
thread that is at the top of the queue will start its transmission and return to subroutine
repeat until queue is empty}