Sample Answer:

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

The number of threads used in the program will be the same number as the number o flows to be simulated given in the input file. Each thread will have a set of characteristics that will define its flow number, arrival time, transmission time and priority.

(2) Do the threads work independently? Or is there an overall “controller” thread?

Each of the threads will work independently of each other and will depend on the monitor to ensure that only one flow will be executing on the interface at one time. Within the thread function that will be created each thread will call a function in the monitor (i.e. request_interface( ) to attempt to run on the interface).

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

There will be a mutex around a function that is modifying the data structures that contain the flow structures to ensure that only one flows is removed/added from the waiting queue at one time.

There will also be a mutex at the beginning of the request_interface( ) function in the monitor to ensure that flows that are waiting to use the interface can be put into the waiting queue and begin waiting on the conditional variable.

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

The main thread will be idle and will wait for all the threads to run on the interface and then terminate.

(5) How are you going to represent flows? What type of data structures will you use?

Each flow is going to be represented by a structure which contains the information regarding each flow's number, arrival time, transmission time and priority.

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

There will be a mutex lock surrounding the data structure so that only one flow can be appended to the list of waiting flows at one time. I am assuming that the flow structures will not need to be modified.

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

(a)Describe the condition that the convar will represent
(b)Which mutex is associated with the convar? Why?
(c)What operation should be performed once pthread_cond_wait( ) has been unblocked and reaquired the mutex?

There will be one convar used in this program. The condition the convar will represent is the state of the interface, Available/Unavailable. This convar is associated with the mutex that will guard the function request_interface( ) to ensure that only one flow can attempt to use the interface at one time. Once the pthread_cond_wait( ) has been unblocked the next process attempting to request_interface( ) should be able to acquire the mutex.

(8) In 25 lines or less, briefly sketch the overall algorithm you will use.

```
Main( ){
    for each thread in file create thread to execute thread_control;
}
thread_control( ){
        sleep until arrival time
        populate thread structures
        request_interface( *flow_struct)
        sleep for burst amount
        release_interface( );
}
monitor MFS( ){
        convar available;
        mutex mutex1, mutex2
        *flow_struct waiting_flows[Max_flows];
        void request_interface(*flow_struct flow){
                lock mutex1
                if (interface available) set interface to unavailable run flow
                lock mutex2
                add flow to the queue of waiting flows and sort it by priority rules
                unlock mutex 2
                while(interface not available and the only waiting flow is the current flow)
                        available.wait( ) //release mutex 1 to allow next flow arriving to
                                come in
                take current process from queue and execute and resort the queue }
        void release_interface(){
                set interface to available
                available.signal( )}
}
```