**CSE321 Realtime and Embedded System Fall 2013**

How to solve the AFLIC problem?

1. We have planes taking off. Design a structure for storing the properties of the flights taking off. (Use struct as illustrated in thread2.c?)
2. Define a table (array) to define multiple flights taking off. Where is an example? In thread2.c
3. How to define the life cycle of the flights taking off? Using a finite state machine.
4. How to implement the finite state machine? Using a switch statement.
5. Where is this switch statement? Inside the workload code of the flight that is taking off. Lets call this code “takeoff”
6. Can you give me an example of “workload” function? It is in thread2.c as the function that each thread executes.
7. How to implement multiple flights taking off? Create multiple threads in a loop.
8. Use printf to print the operation of the flight in each state?
9. How to simulate and protect runways? Use mutexes? Then the flights are automatically queued up. For simplicity assume both landing and taking off flights are queued up. Where can find a mutex example?

Can you transform the same to landing flights? “landing” will be function. You need a separate state diagram.

 50% of the work is done. Submit it. “takeoff” code without “landing” code will get a zero.

 Document your design.

 ================================================================================

 Now for simulating the collision: there are many ways of doing this.

 An approach: let an independent thread generate a random delay, and set a flag for collision between two randomly chosen flights. Where could this information be stored? In the data structure “struct” meant for the flights. Then this information is sensed /checked by the flights in critical states and appropriate actions taken.

Don’t forget design documentation that contains the state diagrams for the two different types of flights.

Don’t forget a REDAME file that tells a user how to run and evaluate your program.

----Ask Brian. Ask Bruce. Ask Jing. Don’t be afraid. Make good use of recitation and office hours.

---- We will post the grading guidelines soon. (takeoff, landing, runway, collision, state diagram design, overall style and design, makefile, readme )