DEPARTMENT OF COMPUTER & INFORMATION SYSTEMS ENGINEERING
BACHELORS IN COMPUTER SYSTEMS ENGINEERING

Complex Engineering Problem

Course Code: CS 329, Course Title: Operating Systems
CLO: CLO 2, Taxonomy Level: C3

Problem Statement

Implement any one of the given classical synchronization problems using *any* appropriate system and application programming tools considering all the conditions and constraints specified in the problem statement.

Attributes

The CEP possesses the following attributes:

- Depth of analysis is required.
- Depth of knowledge is required.
- The problem encompasses familiar issues frequently encountered in system programming.

Deliverables

Students may make a group of at most four for this complex engineering activity. They are supposed to:

- Provide at least three test cases to verify the correctness of solution for the problem.
- Submit a report containing the code, the test cases and the execution snapshots.
- Present the running code to the course instructor when asked to do so.

CEP Rubric

Quality of the code			
0	1	2	3
The code is wrong.	The code is incomplete and unacceptable.	The code is incomplete.	The code is complete.
Test cases			
0	1	2	3
Test cases have not been furnished.	Test cases have been furnished but are inadequate and unacceptable.	Test cases have been furnished but are inadequate.	Test cases have been furnished and are adequate.
Genuine effort			
0	1	2	
The code is copied.	Parts of the code are copied.	The group has genuinely created the code.	
Report			
0	1	2	
The submitted report is unfit to be graded.	The report is partially acceptable.	The report is complete and concise.	

Problems

Consider the statements of the following classical synchronization problems of operating systems:

1. Cigarette smokers' problem

Four threads are involved: an agent and three smokers. The smokers loop forever, first waiting for ingredients, then making and smoking cigarettes. The ingredients are tobacco, paper, and matches. We assume that the agent has an infinite supply of all three ingredients, and each smoker has an infinite supply of one of the ingredients; that is, one smoker has matches, another has paper, and the third has tobacco.

The agent repeatedly chooses two different ingredients at random and makes them available to the smokers. Depending on which ingredients are chosen, the smoker with the complementary ingredient should pick up both resources and proceed. For example, if the agent puts out tobacco and paper, the smoker with the matches should pick up both ingredients, make a cigarette, and then signal the agent. To explain the premise, the agent represents an operating system that allocates resources, and the smokers represent applications that need resources.

The problem is to make sure that if resources are available that would allow one or more applications to proceed, those applications should be woken up. Conversely, we want to avoid waking an application if it cannot proceed.

The restriction imposed on the system is you are not allowed to modify the agent code. If the agent represents an operating system, it makes sense to assume that you don't want to modify it every time a new application comes along. So the abilities of the agent remain as defined in the problem.

2. The barbershop problem

A barbershop consists of a waiting room with n chairs, and the barber room containing the barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy, but chairs are available, then the customer sits in one of the free chairs. If the barber is asleep, the
customer wakes up the barber. Write a program to coordinate the barber and the customers.

To make the problem a little more concrete, the following information is added:

- Customer threads should invoke a function named getHairCut.
- If a customer thread arrives when the shop is full, it can invoke balk, which does not return.
- Barber threads should invoke cutHair.
- When the barber invokes cutHair, there should be exactly one thread invoking getHairCut concurrently.