CSE325-Operating Systems
Project Descriptions

1. **Project Title:** Seeking Tutor Problem

   Project Description:

   The computer science department of EWU runs a programming club to help undergraduate students with their programming assignments. The club has a coordinator and several tutors to assist the students. The waiting area of the center has several chairs. Initially, all the chairs are empty. The coordinator is waiting for the students to arrive. The tutors are either waiting for the coordinator to notify that there are students waiting or they are busy tutoring. The tutoring area is different from the waiting area. A student, while programming for his project, decides to go to a club to get help from a tutor. After arriving at the club, the student sits in an empty chair in the waiting area and waits to be called for tutoring. If no chairs are available, the student will go back to programming and come back to the club later. Once a student arrives, the coordinator queues the student based on the student's priority, and then the coordinator notifies an idle tutor. A tutor, once woken up, finds the student with the highest priority and begins tutoring. A tutor after helping a student, waits for the next student. A student after receiving help from a tutor goes back to programming. The priority of a student is based on the number of times the student has visited the club. A student visiting the club for the first time gets the highest priority. In general, a student visiting for the ith time has a priority higher than the priority of the student visiting for the kth time for any k > i. If two students have the same priority, then the student who came first has a higher priority.

   Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the coordinator, tutors, and the students. The total number of students, the number of tutors, the number of chairs, and the number of times a student seeks a tutor's help are passed as command line arguments. Once a student thread takes the required number of help from the tutors, it should terminate. Once all the student threads are terminated, the tutor threads, the coordinator thread, and the main program should be terminated. Your program should work for any number of students, tutors, chairs and help sought. Allocate memory for data structures dynamically based on the input parameter(s).

2. **Project Title:** Burger Buddies Problem

   Project Description:

Design, implement and test a solution for the IPC problem specified below. Suppose we have the following scenario: Cooks, Cashiers, and Customers are each modeled as a thread. Cashiers sleep until a customer is present. A Customer approaching a cashier can start the order process. A Customer cannot order until the cashier is ready. Once the order is placed, a cashier has to get a burger from the rack. If a burger is not available, a cashier must wait until one is made. The cook will always make burgers and place them on the rack. The cook will wait if the rack is full. There are NO synchronization constraints for a cashier presenting food to the customer. Implement a (concurrent multi-threaded) solution to solve the problem and test it thoroughly.

3. **Project Title:** Amherst Candy Factory Problem

   Project Description:

   Amherst candy factory is preparing up for Halloween and has implemented an assembly line to ramp up Halloween candy production. The assembly line will be implemented using a bounded buffer producer and consumers. The factory contains two types of worker threads: producers and consumers. Each producer thread produces a certain type of candy, while each consumer creates boxes of assorted candies using ones produced by producers. After producing each candy, the producer deposits into the bounded buffer. If the buffer is full, it must wait until one slot becomes available. A consumer extracts candy from the bounded buffer, one at a time, and when *i* candy items have been extracted, it fills up a box of assorted candy.

   Implement a solution that synchronizes the activities of the producers, and the consumers. The total number of producers, consumers and total number of types of candies are passed as command line arguments. Allocate memory for data structures dynamically based on the input parameter(s).

4. **Project Title**: Movie Ratings Problem

   Project Description:

   Rotten tomatoes is a recommendation website for quality entertainment. The viewers can look for the ratings associated with every movie or series. A set of processes concurrently search in the movie database for keywords. Given a keyword (like dragon), a search process reads all the movie records, identifies only those movies whose descriptions contain the keyword, and prints these movies (names, directors, and release dates only) sorted in the decreasing order of the popularity ratings. In order that the database server does not become overloaded by many search processes, allow at most five processes at any instant. Use a counting semaphore to implement this

restriction. When a sixth (or seventh or...) process attempts to make a search, it has to wait until one or more running search processes finish working with the database and signal the semaphore. Implement a synchronization method to solve the problem.

5. **Project Title**: Dental Clinic Problem

   Project Description:

   A fresh graduate from a dental school opened his own dental clinic and started to make a living. This dental clinic has only one dentist, one dental chair, and n chairs for waiting for patients if there are any to sit on the chair. Initially, all the chairs are empty. If there is no patient, the dentist sleeps in his own chair. When a patient arrives, he has to wake up the dentist. If there are many patients and the dentist is attending a patient, then the remaining patients either wait if there are empty chairs in the waiting room or they leave if no chairs are empty. Implement a synchronization method to solve the problem.

   Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the dentist, and the patients. The total number of patients, the number of chairs are passed as command line arguments. Once a patient thread receives treatment from the dentist, it should terminate. Once all the patient threads are terminated, the dentist thread and the main program should be terminated. Your program should work for any number of patients and chairs. Allocate memory for data structures dynamically based on the input parameter(s).

6. **Project Title**: The Sandwich Problem

   Project Description:

   There are four students sitting in a room: three juniors and one senior. Each of the juniors will make a sandwich and eat it. To make a sandwich requires bread, cheese, and sausage. Each junior has one of the three items, that is, one has bread, another has cheese and the third has sausage. The senior has infinite supply of all three items. The senior places two of the three items on the table, and the junior that has the third item, makes the sandwich. However, the other two juniors cannot make a sandwich because they do not have the third required item. The junior who has the remaining ingredient then makes and eats the sandwich, signaling the senior on completion. The senior then puts out another two of the three ingredients and the cycle repeats.

   Implement a synchronization method to solve the problem. You must output the progress of the processes. E.g. when the senior places the two ingredients you should print that information. When a junior gets to make a sandwich, output which

junior got the ingredients and so-on. Show the run of your processes for a while until around 10 sandwiches are made.

7. **Project Title:** FIFA World Cup

   Project Description:

   For the FIFA World Cup 2006, a fly-over has been constructed between the hotel where the teams are staying and the stadium. This fly-over will be used by the German team and the Italian team in the upcoming semifinal on Tuesday. A tram car is used to cross this fly-over, but it seats only four people, and must always carry a full load. We cannot put three Italians and one German in the same tramcar, because the Italians would be in majority and might try to intimidate the German. Similarly, we cannot put three Germans in the same tram-car with one Italian. All other combinations are safe.

   Implement a synchronization method to solve the problem. You must output the progress of the processes. E.g. When a player arrives you should output who arrived and also print the total number of Germans and Italians waiting to board the tram. When a tram leaves you should output that as well.

8. **Project Title:** Study tour in a museum

   Project Description:

   A museum arranged a day long program for some school children in which students can participate in various activities. In one room, a 3D short movie has been shown and so 3D glasses are needed. Also, after they have watched the movie, they need to return the 3D glasses. However, only a limited number of 3D glasses are available. So, all students may not be able to watch it at the same time. Some students have to wait unless they get 3D glasses. If the waiting area is full, the other students have to leave.

   Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the students. The total number of students, the number of 3D glasses are passed as command line arguments. Once a student thread finish watching the movie, it should terminate. Once all the student threads are terminated, the main program should be terminated. Your program should work for any number of students and 3D glasses. Allocate memory for data structures dynamically based on the input parameter(s).

9. **Project Title:** Movie Ticketing System

Project Description:

Star Cineplex is the first multiplex cinema theatre in Bangladesh. It also gives its viewers the facility to check available tickets and book shows via online ticket booking system. When the program is run, it creates a certain number of threads that attempt to sell all available tickets. However, the program needs to make sure that while booking the same show by various users, the total number of available seats are error free and there is not any data loss.

Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the users. The total number of users, the number of tickets, and number of shows are passed as command line arguments. Once a user thread finish booking a show, it should terminate. Once all the user threads are terminated, the main program should be terminated. Your program should work for any number of users, tickets, and shows. Allocate memory for data structures dynamically based on the input parameter(s).