**PROJECT PROPOSAL**

(Operating System)

**Group Members:**

1. Seemi Urooj (21K-3345)
2. Anfaal Abdul Hai (21K-4879)
3. **Title: Dining Philosophers Problem** (System Call).
4. **Introduction**

The operating system is a program that links the user and the computer system. This operating system must be capable of controlling resource usage. In the process of designing the operating system, there is a common foundation called concurrency. Concurrent processes are when the processes work at the same time. This is called the multitasking operating system. Concurrent processes can be completely independent of the other but can also interact with each other.

Processes that require synchronization to interact properly controlled. However, the concurrent processes that interact, there are some problems to be solved such as deadlock and synchronization. One of the classic problems that can illustrate the problem is the Dining Philosophers Problem.Dining Philosophers Problem can be illustrated as follows; there are five philosophers who would eat. On the table was reserved five chopsticks. If philosophers really hungry, then it will take two chopsticks, which is in the right and left hands. However, sometimes only one course takes chopsticks. If there are philosophers who took two chopsticks, then there are philosophers who have to wait until the chopsticks are placed back. Inside this problem, there is the possibility of deadlock, a condition in which two or more processes can not continue execution.

1. **Methodology**

The philosophers are sitting around a round table, and there is a big bowl of spaghetti at the center of the table. There are five forks placed around the table in between the philosophers. When a philosopher, who is mostly in the thinking business gets hungry, he grabs the two forks to his immediate left and right and dead-set on getting a meal; he gorges on the spaghetti with them. Once he is full, the forks are placed back, and he goes into his mental world again. The problem usually omits an important fact that a philosopher never talks to another philosopher. The typically projected scenario is that if all the philosophers grab their fork on their left simultaneously none of them will be able to grab the fork on their right. Moreover, with their one-track mindset, they will forever keep waiting for the fork on their right to come back on the table.

The basic idea behind the scenario is that if a concurrent activity always does what seems best for itself, the result can be chaos. Is there a solution to the Dining Philosopher Problem? The scenario was posed not for a solution but to illustrate a basic problem if the traditional programming approach is applied to concurrent systems. The problem itself crops up in the concurrent systems, and the design decisions should be aware of this, and that is what we have to solve. Any set of concurrent programming techniques that we use is expected at the basic level to offer us features that can be used to deal with the Dining Philosophers problem in some way.  
Assume that we have the simple task of writing some important information into two files on the disk. However, these files are shared by other programs as well. Therefore we use the following strategy to update the files:  
Lock A  
Lock B  
Write information to A and B  
Release the locks

This obvious coding can result in deadlocks if other tasks are also writing to these files. For example, if another task locks B first, then locks A, and if both tasks try to do their job at the same time – dead-lock occurs. My task would lock A, the other task would lock B, then my task would wait indefinitely to lock B while the other task waits indefinitely to lock A. This is a simple scenario, and easy to find out. However, you can have a bit more involved case where task A can wait for a lock held by task B which is waiting for a lock held by task C which is waiting for a lock held by task A. A circular wait a deadlock results. This is a Dining Philosophers model.

In the above code fragment, one could resort to locking the files one at a time for modification. Then the problem would disappear. However, there are times when requirements dictate that it has to be locked more than one resource before updating them.

1. **Conclusion:**

Dining Philosophers Problem is one of the classic issues in the operating systems. Dining Philosophers Problem can be described as follows; there are five philosophers who want to eat. There are five chopsticks on the table. Each philosopher must use two chopsticks if he would like to eat the spaghetti. If philosophers really hungry, then it will take two chopsticks, which is in the right and left hands. If there are philosophers who took two chopsticks, then there are philosophers who have to wait until the chopsticks are placed back. Inside this problem there is the possibility of deadlock.