***LAB MANUAL # 5:***

***Roll\_no 068464***

***Submitted \_by Nabila Naz***

***Submitted To Sir Kamran .***

***Bs (IT) 5th(sem) Morning***

# **CC-311 Operating Systems LAB**

**“synchronization through semaphores “**

**Q1. Write a program to achieve synchronization between multiple threads. The threads try to acquire a resource that has two instances.**

***Viva Questions:***

**Q1. What is the initial value of the semaphore variable?**

* **. The initial value of the semaphore variable is specified as a parameter to the**sem\_init()**function. In this program, the initial value of the semaphore is set to 1, indicating that the resource is available for acquisition.**

**Q2. Why we use *pthread\_join()* function in the above program?**

* **The** pthread\_join()**function is used to wait for a thread to finish its execution. In this program, we use**pthread\_join() **to wait for all the threads to finish their execution before destroying the semaphores and returning from the main function.**

**Q3. Why is the fourth parameter in *pthread\_create() NULL*?**

* **The fourth parameter in the**pthread\_create() f**unction is for thread attributes. In this program, we pass**NULL **as the thread attributes, which means that we are using the default thread attributes.**

**Q4. What is the significance of using *sleep(1)* function in the functions *fun1()* and *fun2()*?**

* **The**sleep(1)**function is used in the**fun1()**and** fun2()**functions to simulate some work being done with the resource. In a real-world program, you would replace this with your own code that uses the resource. The s**leep(1**) function is used here to ensure that the threads do not release the resource too quickly, allowing us to see the synchronization in action.**

**Q5. How to use counting semaphores?**

* **. Counting semaphores are a type of semaphore that can keep track of a count of resources, rather than just indicating whether a resource is available or not. To use a counting semaphore, you would initialize it with an initial count using the**sem\_init() fu**nction, and then use**sem\_wait() **to decrement the count and** sem\_post() **to increment the count. When the count reaches zero, further calls to** sem\_wait()**will block until the count is increased again. Here's an example of how to use a counting semaphore:**