Exercise 9 - Dining philosopher using semaphore and Reader writer problem with some constraints:

1. **Code:**

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
#define rep(i,k,n) for(int i=k;i<n;i++)
sem_t chps[5];
void * philos(void * n)
{
    int p=*(int *)n;
    sem_wait(&chps[p]);
    printf("Philosopher %d picks left chopsticks\n",p);
    sem_wait(&chps[(p+1)%5]);
    printf("Philosopher %d picks the right chopsticks\n",p);
    printf("Philosopher %d starts eating\n",p);
    sleep(1);
    printf("Philosopher %d done with eating\n",p);
    sem_post(&chps[(p+1)%5]);
    sem_post(&chps[p]);
}
int main()
 {
         int a[5];
         pthread_t t[5];
         rep(i,0,5)
         sem_init(&chps[i],0,1);
         rep(i,0,5){
                 a[i]=i;
                 pthread_create(&t[i],NULL,philos,(void *)&a[i]);
            }
         rep(i,0,5)
            pthread_join(t[i],NULL);
 }
```

Output:

```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab9$
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab9$
Philosopher 0 picks left chopsticks
Philosopher 1 picks left chopsticks
Philosopher 1 picks the right chopsticks
Philosopher 1 starts eating
Philosopher 3 picks left chopsticks
Philosopher 3 picks the right chopsticks
Philosopher 3 starts eating
Philosopher 1 done with eating
Philosopher 2 picks left chopsticks
Philosopher 0 picks the right chopsticks
Philosopher 0 starts eating
Philosopher 3 done with eating
Philosopher 4 picks left chopsticks
Philosopher 2 picks the right chopsticks
Philosopher 2 starts eating
Philosopher 0 done with eating
Philosopher 4 picks the right chopsticks
Philosopher 4 starts eating
Philosopher 2 done with eating
Philosopher 4 done with eating
```

Code:

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>

sem_t S,P;
pthread_mutex_t mutex;
pthread_mutex_t mutexwriter;
int cnt = 1;
int numreader = 0;
int numwriter = 0;

void *writer(void *wno)
{

    pthread_mutex_lock(&mutex);
    pthread_mutex_lock(&mutexwriter);
```

```
    sleep(2);
    printf("Writer %d writing\n",(*((int *)wno)));
    printf("Writer %d finished\n",(*((int *)wno)));
    sem_post(&S);
    pthread_mutex_unlock(&mutexwriter);
    pthread_mutex_unlock(&mutex);

}
void *reader(void *rno)
{
    sem_wait(&S);

    pthread_mutex_lock(&mutex);
    numreader++;
    pthread_mutex_unlock(&mutex);
    sleep(2);
    printf("Reader %d: read\n",*((int *)rno));
    pthread_mutex_lock(&mutex);
    numreader--;
    pthread_mutex_unlock(&mutex);

    pthread_mutex_lock(&mutex);
    if(numreader == 5) {
        sem_post(&S);
    }
    pthread_mutex_unlock(&mutex);
    // pthread_mutex_unlock(&mutexwriter);
        sem_post(&S);


}

int main()
{

    pthread_t read[10],write[7];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&S,0,0);

    int a[10] = {1,2,3,4,5,6,7,8,9,10};

    for(int i = 0; i < 7; i++) {
        pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
```

```
    }
    for(int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }
    for(int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }

    pthread_mutex_destroy(&mutex);
    sem_destroy(&S);

    return 0;

}
```

Output

```
shubhangi@Shubhi:/mnt/e/VIT/4thsem/OS/lab/linuxpractice/20bce1161/lab9$ ./rwoutput
Writer 3 writing
Writer 3 finished
Writer 2 writing
Writer 2 finished
Reader 2: read
Reader 1: read
Writer 1 writing
Writer 1 finished
Writer 5 writing
Writer 5 finished
Reader 6: read
Reader 3: read
Reader 4: read
Writer 4 writing
Writer 4 finished
Reader 5: read
Reader 7: read
Reader 8: read
Reader 10: read
Reader 9: read
```