Assignment Number 04

Name: Mihir Unmesh Patil

Roll NO: TYCOC213

Batch: C/ C-3

CODE:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#include <sched.h>
#define MAX 50

int sleepMod = 5;
int readCount = 0;
int isSync = 1; // Default: Synchronous mode

sem_t readAccess, bookAccess;

void *reader_func(void *);
void *writer_func(void *);

int main() {
   srand(time(0));
   int readers, writers, mode;

   printf("Choose mode: 1 for Synchronous, 2 for Asynchronous: ");
   scanf("%d", &mode);
   isSync = (mode == 1) ? 1 : 0;

   printf("Number of readers (max 50): ");
   scanf("%d", &readers);
   printf("Number of writers (max 50): ");
   scanf("%d", &writers);

   if (readers > 5) sleepMod = readers;

   pthread_t readers_t[MAX], writers_t[MAX];
   sem_init(&readAccess, 0, 1);
   sem_init(&bookAccess, 0, 1);

   int i;
   for (i = 0; i < readers; i++)
      pthread_create(&readers_t[i], NULL, reader_func, &i);
   for (i = 0; i < writers; i++)
      pthread_create(&writers_t[i], NULL, writer_func, &i);

   for (i = 0; i < writers; i++)
      pthread_join(writers_t[i], NULL);
   for (i = 0; i < readers; i++)
      pthread_join(readers_t[i], NULL);
```

```
    sem_destroy(&readAccess);
    sem_destroy(&bookAccess);
    return 0;
}

void *reader_func(void *r) {
    int rNo = *((int *)r) + 1;
    printf("\n Reader %d: wanting to read",
rNo);
    sleep(rand() % sleepMod);

    if (isSync)
        sem_wait(&readAccess);
    else
        sched_yield();

    readCount++;
    if (readCount == 1)
        sem_wait(&bookAccess);

    printf("\n Reader %d: reading", rNo);
    if (isSync)
        sem_post(&readAccess);
    sleep(rand() % sleepMod);

    if (isSync)
        sem_wait(&readAccess);

    readCount--;
    printf("\n Reader %d: leaving reading",
rNo);
    sleep(rand() % sleepMod);

    if (readCount == 0)
        sem_post(&bookAccess);

    if (isSync)
        sem_post(&readAccess);
    printf("\n Reader %d: finished", rNo);
    sleep(rand() % sleepMod);

    pthread_exit(0);
}

void *writer_func(void *w) {
    int wNo = *((int *)w) + 1;
    printf("\n Writer %d: wanting to write",
wNo);
    sleep(rand() % sleepMod);

    if (isSync)
        sem_wait(&bookAccess);
    else
        sched_yield();

    printf("\n Writer %d: writing", wNo);
    sleep(rand() % sleepMod);

    printf("\n Writer %d: leaving writing", wNo);
    sleep(rand() % sleepMod);

    if (isSync)
        sem_post(&bookAccess);
```

```
    printf("\n Writer %d: finished", wNo);
    sleep(rand() % sleepMod);

    pthread_exit(0);
}
```

Output: