Producer Consumer Problem :

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<pthread.h>
#include<unistd.h>
#include<semaphore.h>
#define BUFFER_SIZE 5

sem_t empty;//  for buffer empty
sem_t full;//  for buffer full
pthread_mutex_t mutual;// used for mutual exclusion
int buffer[BUFFER_SIZE];

void *producer(void *arg)
{
   int item,i=0;
   while(1)
   {
      item=rand()%100;
      sem_wait(&empty);
      pthread_mutex_lock(&mutual);
      sleep(1);
      printf("\n%d item is inserting in buffer at posn %d\n\n",item,i);
      buffer[i++]=item;
      if(i==BUFFER_SIZE)
            i=0;
      pthread_mutex_unlock(&mutual);
      sem_post(&full);
      sleep(1);
   }
}
```

```
void *consumer(void *args)
{
   int item,i=0;
   while(1)
   {   sem_wait(&full);
       pthread_mutex_lock(&mutual);
            sleep(1);
       item=buffer[i];
       printf("\n%d item is consumed from buffer from position %d\n",item,i);
        i++;
        if(i==BUFFER_SIZE)
i=0;
       pthread_mutex_unlock(&mutual);
       sem_post(&empty);
       sleep(1);
   }
}
int main()
{
   sem_init(&empty,0,BUFFER_SIZE);
   sem_init(&full,0,0);
   pthread_mutex_init(&mutual,NULL);
   pthread_t pid,cid;
   pthread_create(&pid,NULL,producer,NULL);
   pthread_create(&cid,NULL,consumer,NULL);
   pthread_join(pid,NULL);
   pthread_join(cid,NULL);
   return 0;
}
```

OUTPUT :

main.c

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<pthread.h>

pthread_t philosopher[5];
pthread_mutex_t spoon[5];

void *func(int n)
{
 printf("Philosopher %d is thinking\n",n);
 pthread_mutex_lock(&spoon[n]);
 pthread_mutex_lock(&spoon[(n+1)%5]);
```

input

```
Philosopher 4 is eating
Philosopher 2 is thinking
Philosopher 2 is eating
Philosopher 1 is thinking
Philosopher 0 is thinking
Philosopher 3 is thinking
Philosopher 4 finished eating
Philosopher 2 finished eating
Philosopher 3 is eating
Philosopher 1 is eating
Philosopher 3 finished eating
Philosopher 1 finished eating
Philosopher 0 is eating
Philosopher 0 finished eating


...Program finished with exit code 0
```

Reader Writer Problem :

```
//inclusion of header files
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<semaphore.h>
int rc=0;              //initialise counter
sem_t mutex,wc;        //declaring semaphore object
void *reader(void *arg) //function defination
{
 int i;
i=(int)arg;
sem_wait(&mutex);      //function call
rc=rc+1;
printf("\nReader %d is waiting ",i);
if(rc==1)
```

```
sem_wait(&wc);         //function call
sem_post(&mutex);      //function call
printf("\n Reader %dis in critical sec ",i);
sleep(2);
printf("\nReader %d is exiting",i);
sem_wait(&mutex);      //function call
rc=rc--;
if(rc==0)
sem_post(&wc);         //function call
sem_post(&mutex);      //function call
pthread_exit(NULL);
}
void *writer(void *arg) //function defination
{
sem_wait(&wc);          //function call
printf("\n writer %d is in c.s ",(int *)arg);
sleep(2);
printf("\n writer %d is existing ",(int *)arg);
sem_post(&wc);         //function call
pthread_exit(NULL);
}
int main()
{
int i;
pthread_t th[10];
sem_init(&mutex,0,1);  //initialise semaphore
sem_init(&wc,0,1);    //iniialise semaphore
for(i=0;i<5;i++)
pthread_create(&th[i],NULL,reader,(void*)i);//creating thread
for(i=5;i<10;i++)
pthread_create(&th[i],NULL,writer,(void*)i);//creating thread
pthread_join(th[4],NULL);
pthread_join(th[9],NULL);
```

pthread_join(th[3],NULL);

pthread_join(th[0],NULL);

pthread_join(th[5],NULL);

pthread_join(th[6],NULL);

pthread_join(th[7],NULL);

pthread_join(th[2],NULL);

pthread_join(th[8],NULL);

pthread_join(th[1],NULL);

return(0);

}

OUTPUT :

main.c

```
1
2  //inclusion of header files
3  #include<stdio.h>
4  #include<stdlib.h>
5  #include<unistd.h>
6  #include<pthread.h>
7  #include<semaphore.h>
8  int rc=0;                //initialise counter
9  sem_t mutex,wc;          //declaring semaphore object
10 void *reader(void *arg) //function defination
11 {
12  int i;
13 i=(int)arg;
14 sem_wait(&mutex);        //function call
```

input

```
   48 | pthread_create(&th[i],NULL,writer,(void*)i);//creating thread
      |                                    ^

Reader 0 is waiting
 Reader 0is in critical sec
Reader 1 is waiting
 Reader 1is in critical sec
Reader 2 is waiting
 Reader 2is in critical sec
Reader 3 is waiting
 Reader 3is in critical sec
Reader 4 is waiting
 Reader 4is in critical sec
Reader 1 is exiting
Reader 0 is exiting
Reader 2 is exiting
Reader 3 is exiting
```

Dinning Philosopher Problem :

```
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<pthread.h>

pthread_t philosopher[5];
pthread_mutex_t spoon[5];

void *func(int n)
{
 printf("Philosopher %d is thinking\n",n);
 pthread_mutex_lock(&spoon[n]);
 pthread_mutex_lock(&spoon[(n+1)%5]);
 printf("Philosopher %d is eating\n",n);
 sleep(3);
 pthread_mutex_unlock(&spoon[n]);
 pthread_mutex_unlock(&spoon[(n+1)%5]);
 printf("Philosopher %d finished eating\n",n);
 return(NULL);
}

int main()
{
 int i;
 for(i=0;i<5;i++)
  pthread_mutex_init(&spoon[i],NULL);
 for(i=0;i<5;i++)
  pthread_create(&philosopher[i],NULL,(void *)func,(void *)i);
 for(i=0;i<5;i++)
  pthread_join(philosopher[i],NULL);
```

```
 for(i=0;i<5;i++)
  pthread_mutex_destroy(&spoon[i]);
 return 0;
}
```

OUTPUT :

main.c

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<pthread.h>

pthread_t philosopher[5];
pthread_mutex_t spoon[5];

void *func(int n)
{
 printf("Philosopher %d is thinking\n",n);
 pthread_mutex_lock(&spoon[n]);
 pthread_mutex_lock(&spoon[(n+1)%5]);
```

input

```
Philosopher 0 is thinking
Philosopher 0 is eating
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 2 is eating
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 0 finished eating
Philosopher 1 is eating
Philosopher 2 finished eating
Philosopher 4 is eating
Philosopher 1 finished eating
Philosopher 4 finished eating
Philosopher 3 is eating
Philosopher 3 finished eating


...Program finished with exit code 0
```