PROGRAM 4:

(i) Reader Writer Problem

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<pthread.h>
#include<sys/fcntl.h>
#include<stdlib.h>
#include<semaphore.h>
int readcount=0;
int count=0;
void *read_task();
void *write_task();
sem_t mutex,wrt;
int main()
{
    int i=0;
    pthread_t readid[10];
    pthread_t writeid;
    sem_init(&mutex,0,1);
    sem_init(&wrt,0,1);
    for(i=0;i<10;i++){
       pthread_create(&readid[i],NULL,read_task,NULL);
       if(i==4)
                  pthread_create(&writeid,NULL,write_task,NULL);
    }
    for(i=0;i<10;i++)
         pthread_join(readid[i],NULL);
    pthread_join(writeid,NULL);
    sem_destroy(&mutex);
    sem_destroy(&wrt);
}
void *read_task()
{
  int fd;
  sem_wait(&mutex);
  readcount++;
```

```
  if(readcount==1)
    sem_wait(&wrt);
  char sentence[50];
  fd=open("a.txt",0);
  read(fd,sentence,30);
  printf("Reader %d reading\n",count++);
  printf("%s\n",sentence);
  sem_post(&mutex);
  close(fd);
  sem_wait(&mutex);
  readcount--;
  sleep(5);
  if(readcount == 0)
    sem_post(&wrt);
  sem_post(&mutex);
}

void *write_task()
{
  int fd;
  char sentence[50]="RV College of Engineering";
  sem_wait(&wrt);
  fd = open("a.txt",1);
  lseek(fd,SEEK_SET,30);
  write(fd,sentence,50);
  printf("Writer is writting\n");
  close(fd);
  sem_post(&wrt);
}
```

(i) Producer-Consumer Problem

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

```
#define MAX 5
int in=0, out=0;
void *producer();
void *consumer();
sem_t empty,full;
int main()
{
    int i=0;
    pthread_t prod_id, cons_id;
    sem_init(&empty,0,MAX);
    sem_init(&full,0,0);

    pthread_create(&prod_id,NULL,producer,NULL);
    sleep(2);
    pthread_create(&cons_id,NULL,consumer,NULL);

    pthread_join(prod_id,NULL);
    pthread_join(cons_id,NULL);
    sem_destroy(&empty);
    sem_destroy(&full);
}
void *producer()
{
   int i,sleep_count;
   for(i=0;i<20;i++)
   {
      sem_wait(&empty);
        printf("Producer produced at %d\n",in);
        in = (in + 1) % MAX;
        //sleep_count=rand()%3;
        sleep(3);
      sem_post(&full);
   }
}

void *consumer()
{
   int i,sleep_count;
   for(i=0;i<20;i++)
   {
```

```
        sem_wait(&full);
          printf("\tConsumer consumed from %d\n",out);
          out = (out + 1) % MAX;
          //sleep_count=rand()%3;
          sleep(2);
        sem_post(&empty);
    }
}
```

(iii) Dining Philosopher

```
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
int state[5];
sem_t self[5];
void *philosopher(void *num);
void pickup(int);
void test(int);
void putdown(int);
int phil_no[5]={0,1,2,3,4};
int main()
{
    int i,k;
    pthread_t tid;
    for(i=0;i<5;i++){
        state[i]=0;
        sem_init(&self[i],0,1);
    }
    for(i=0;i<10;i++){
         k= i % 5;
         pthread_create(&tid,NULL,philosopher,&phil_no[k]);
         sleep(1);
    }
    pthread_join(tid,NULL);
    for(i=0;i<5;i++)
    sem_destroy(&self[i]);
```

```
}
void *philosopher(void *num)
{
  int *i=num;
  pickup(*i);
  sleep(10);
  putdown(*i);
  printf("Philosopher %d is putdown\n",*i);

}
void pickup(int i)
{
  state[i]=1;
  test(i);
  if(state[i]!=2)
    sem_wait(&self[i]);
}
void test(int i)
{
   printf("states in test are %d:%d %d:%d
%d:%d\n",i,state[i],(i+1)%5,state[(i+1)%5],(i+4)%5,state[(i+4)%5]);
   if(state[i]==1 && state[(i+1)%5] != 2 && state[(i+4)%5] != 2){
        state[i]=2;
        printf("Philosopher %d is eating\n",i);
        sem_post(&self[i]);
   }
}
void putdown(int i)
{
   state[i]=0;
   test((i+1)%5);
   test((i+4)%5);
}
```