```
#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<semaphore.h>

#include <time.h>



void * cat();

void * mice();

int NumBowls[20],num=0,arr[20];

int NumCats=0,NumMice=0;

sem_t numberOfCats,numberOfMice;

pthread_t thread1,thread2,thread3,thread4,thread5;

pthread_mutex_t mutex,catmutex,micemutex;


void * mice()

{

                NumMice=NumMice+1;

                arr[NumMice]=NumMice;

                int i=NumMice;


        sem_wait(&numberOfMice);

        if(NumMice==1){

                                pthread_mutex_lock(&micemutex);

}

        printf("MOUSE %d IS EATING \n",NumMice);

        printf("MOUSE %d IS SLEEPING \n",NumMice);

        sleep(5);

        if(i!=arr[i])
        {
                return 0;
        }
        printf("MOUSE %d WOKE UP AND STARTS EATING \n",NumMice);
        sleep(5);

        printf("MOUSE %d HAS EXECUTED\n",NumMice);

        pthread_mutex_unlock(&micemutex);
}

void * cat()
{
        pthread_mutex_lock(&mutex);
        NumCats=NumCats+1;
        num=num+1;
        printf("CAT %d HAS STARTED ITS EXECUTION \n",NumCats);
        printf("CAT %d IS NOW SLEEPING \n",NumCats);
        sleep(5);

        printf("CAT %d WOKE UP \n",NumCats);
        while(NumMice>0)
        {
        sem_destroy(&numberOfMice);
        printf("MOUSE %d IS DEAD %d \n",NumMice);
        arr[NumMice]=-1;
        NumMice=NumMice-1;
        }
        printf("CAT %d IS NOW SLEEPING AGAIN\n",NumCats);
```

```
            sleep(5);


            printf("CAT %d WOKE UP AND STARTS EATING\n",NumCats);
            NumBowls[num]=num;
            printf("CAT %d HAS FINISHED ITS EXECUTION \n",NumCats);
            pthread_mutex_unlock(&mutex);
}


int main()
{   int num=5,x;
            sem_init(&numberOfCats,0,5);
            sem_init(&numberOfMice,0,5);
            pthread_create(&thread1,NULL,cat,NULL);
            sleep(10);
            pthread_create(&thread2,NULL,cat,NULL);
            pthread_create(&thread3,NULL,cat,NULL);
            sleep(10);
            pthread_create(&thread4,NULL,cat,NULL);
            pthread_create(&thread5,NULL,mice,NULL);
            pthread_join(thread1,NULL);
            pthread_join(thread2,NULL);
            pthread_join(thread3,NULL);
            pthread_join(thread4,NULL);
            pthread_join(thread5,NULL);
            }
```