#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#include<semaphore.h>

#include<unistd.h>

#define buffer\_size 10

sem\_t full,empty;

int buffer[buffer\_size];

pthread\_mutex\_t mutex; //daclare mutex variable

void \*producer(void \*p);//function prototype

void \*consumer(void \*p);

void insert\_item(int);

int remove\_item();

int counter;

void initialize()

{

pthread\_mutex\_init(&mutex,NULL);//declaration of mutex

//initialise semaphore variables

sem\_init(&full,1,0);

sem\_init(&empty,1,buffer\_size);

counter=0;

}

int main()

{

int n1,n2,i;

printf("Enter no. of producers you want to create:");

scanf("%d",&n1);

printf("Enter no. of consumers you want to create:");

scanf("%d",&n2);

initialize();

pthread\_t tid1[n1],tid2[n2];//tid==thread id

// create producer thread

for(i=0;i<n1;i++)

pthread\_create(&tid1[i],NULL,producer,NULL);

//create consumer thread

for(i=0;i<n2;i++)

pthread\_create(&tid2[i],NULL,consumer,NULL);

sleep(20);

exit(0);

}

void \*producer(void \*p)

{

int item,waittime;

waittime=rand()%5;//produce random item

sleep(waittime);

item =rand()%10;

sem\_wait(&empty);//producer is allowed to produce data and waiting for his turn

pthread\_mutex\_lock(&mutex);//producer is producing data

printf("\n Producer produced %d item",item);

insert\_item(item);

pthread\_mutex\_unlock(&mutex);//producer have released mutex lock

sem\_post(&full);//producer have increase full slots by 1

}

void \*consumer(void \*p)

{

int item,waittime;

waittime=rand()%10;

sleep(waittime);

sem\_wait(&full); //consumer is trying to consume data

pthread\_mutex\_lock(&mutex);//consumer is consuming a data

item=remove\_item();

printf("\n Consumer consumed %d item",item);

pthread\_mutex\_unlock(&mutex);//consumer have released mutex lock

sem\_post(&empty);//consumer have increased empty slot by 1

}

void insert\_item(int item)

{

buffer[counter++]=item;

}

int remove\_item()

{

return(buffer[--counter]);

}