/\*

Title : Producer -COnsumer

Problem Statement :Thread synchronization and mutual exclusion using mutex. Application to demonstrate: Reader-Writer problem.

\*/

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

#define MaxItems 5 // Maximum items a producer can produce or a consumer can consume

#define BufferSize 5 // Size of the buffer

sem\_t empty;

sem\_t full;

int in = 0;

int out = 0;

int buffer[BufferSize];

pthread\_mutex\_t mutex;

void \*producer(void \*pno)

{

int item;

for(int i = 0; i < MaxItems; i++) {

item = rand(); // Produce an random item

sem\_wait(&empty);

pthread\_mutex\_lock(&mutex);

buffer[in] = item;

printf("Producer %d: Insert Item %d at %d\n", \*((int \*)pno),buffer[in],in);

in = (in+1)%BufferSize;

pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

}

void \*consumer(void \*cno)

{

for(int i = 0; i < MaxItems; i++) {

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

int item = buffer[out];

printf("Consumer %d: Remove Item %d from %d\n",\*((int \*)cno),item, out);

out = (out+1)%BufferSize;

pthread\_mutex\_unlock(&mutex);

sem\_post(&empty);

}

}

int main()

{

pthread\_t pro[5],con[5];

pthread\_mutex\_init(&mutex, NULL);

sem\_init(&empty,0,BufferSize);

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

printf("\nEnter the Number of Producer : ");

int P;

scanf("%d",&P);

printf("\nEnter the Number of Consumer : ");

int C;

scanf("%d",&C);

while(1)

{

for(int i = 0; i < P; i++)

{

pthread\_create(&pro[i], NULL, (void \*)producer, (void \*)&i);

}

for(int i = 0; i < C; i++)

{

pthread\_create(&con[i], NULL, (void \*)consumer, (void \*)&i);

}

for(int i = 0; i < 5; i++)

{

pthread\_join(pro[i], NULL);

}

for(int i = 0; i < 5; i++)

{

pthread\_join(con[i], NULL);

}

}

pthread\_mutex\_destroy(&mutex);

sem\_destroy(&empty);

sem\_destroy(&full);

return 0;

}