Experiment Number: 7

Problem Statement: Producer-Consumer

NAME: **Manomay Jamble**  ROLLNO: 49

CLASS: TY(AI) BATCH: 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PROGRAM:**

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <stdio.h>

#define MaxItems 2 // maximum items a producer can produce or a consumer can consume

#define BufferSize 2 // size of the buffer

sem\_t empty;

sem\_t full;

int in = 0;//index at which producer will put the next data

int out = 0;// index from which the consumer will consume next data

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);//wait() operation to reduce the value by 1

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);//signal() operation

sem\_post(&full);//signal() operation

}

}

void \*consumer(void \*cno)

{

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

sem\_wait(&full);

pthread\_mutex\_lock(&mutex);//same aswait() operation

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);//signal() operation

sem\_post(&empty);//this is the signal() operation to increment the value by 1

}

}

int main()

{

//two producers and consumers

pthread\_t pro[2],con[2];

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

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

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

int a[2] = {1,2}; //just used for numbering the producer and consumer

for(int i = 0; i < 2; i++) {

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

}

for(int i = 0; i < 2; i++) {

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

}

for(int i = 0; i < 2; i++) {

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

}

for(int i = 0; i < 2; i++) {

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

}

pthread\_mutex\_destroy(&mutex);//this is to destroy the mutex to avoid memory leakage

sem\_destroy(&empty);

sem\_destroy(&full);

return 0;

}

**Output:**

