**Experiment no: -7**

Q1. Implement the producer consumer problem using pthreads and mutex operations.

Test Cases:

(a) A producer only produce if buffer is empty and consumer only consume if some contents is in the buffer.

(b) A producer produce an items in the buffer and consumer consumes the last produces item in the buffer.

(c) A producer produces on the last consumed index of the buffer .

Code : -

#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#define N 10

int buffer[N];

int count = 0;

int in = 0;

int out = 0;

pthread\_mutex\_t mutex;

pthread\_cond\_t full;

pthread\_cond\_t empty;

void\* producer(){

    int item;

    while(1){

        scanf("%d",&item);

        pthread\_mutex\_lock(&mutex);

        while(count == N){

            pthread\_cond\_wait(&empty,&mutex);

        }

        buffer[in]=item;

        in = (in + 1)%N;

        count++;

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

        pthread\_cond\_signal(&full);

        pthread\_mutex\_unlock(&mutex);

    }

}

void\* consumer(){

    int item ;

    printf("\n\n");

    while(1){

        pthread\_mutex\_lock(&mutex);

        while(count==0){

            pthread\_cond\_wait(&full,&mutex);

        }

        item = buffer[out];

        out = (out +1 )%N;

        count--;

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

        pthread\_cond\_signal(&empty);

        pthread\_mutex\_unlock(&mutex);

    }

}

int main(){

    pthread\_t t1, t2;

    pthread\_create(&t1,NULL,producer,NULL);

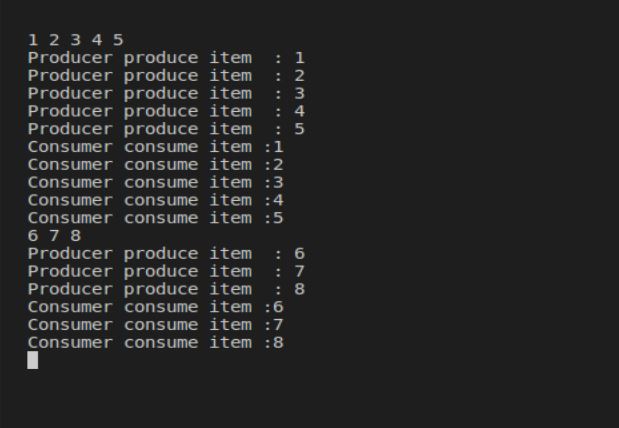
    pthread\_create(&t2,NULL,consumer,NULL);

    pthread\_join(t1,NULL);

    pthread\_join(t2,NULL);

}

Output : -



Q2. Implement the reader writer problem using semaphore and mutex operations to synchronize n readers active in reader section at a same time, and one writer active at a time.

Test Cases:

(a) If n readers are active no writer is allowed to write.

(b) If one writer is writing no other writer should be allowed to read or write on the shared variable

Code:-

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <sys/types.h>

#define NUM\_READERS 4

#define NUM\_WRITERS 1

int shared\_variable = 0;

int readers\_count = 0;

sem\_t mutex;

sem\_t write\_permission;

sem\_t read\_permission;

void \*reader(void \*arg) {

    int id = \*(int \*)arg;

    while (1) {

        sem\_wait(&mutex);

        readers\_count++;

        if (readers\_count == 1) {

            sem\_wait(&write\_permission);

        }

        sem\_post(&mutex);

        printf("Reader %d is reading the shared variable: %d\n", id, shared\_variable);

        sem\_wait(&mutex);

        readers\_count--;

        if (readers\_count == 0) {

            sem\_post(&write\_permission);

        }

        sem\_post(&mutex);

        sleep(10);

    }

}

void \*writer(void \*arg) {

    int id = \*(int \*)arg;

    while (1) {

        sem\_wait(&write\_permission);

        printf("Writer %d is writing the shared variable\n", id);

        shared\_variable++;

        sem\_post(&write\_permission);

        sleep(10);

    }

}

int main() {

    pthread\_t readers[NUM\_READERS], writers[NUM\_WRITERS];

    int i, reader\_ids[NUM\_READERS], writer\_ids[NUM\_WRITERS];

    sem\_init(&mutex, 0, 1);

    sem\_init(&write\_permission, 0, 1);

    sem\_init(&read\_permission, 0, NUM\_READERS);

    for (i = 0; i < NUM\_WRITERS; i++) {

        writer\_ids[i] = i;

        pthread\_create(&writers[i], NULL, writer, (void \*)&writer\_ids[i]);

    }

    for (i = 0; i < NUM\_READERS; i++) {

        reader\_ids[i] = i;

        pthread\_create(&readers[i], NULL, reader, (void \*)&reader\_ids[i]);

    }

    for (i = 0; i < NUM\_READERS; i++) {

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

    }

    for (i = 0; i < NUM\_WRITERS; i++) {

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

    }

    sem\_destroy(&mutex);

    sem\_destroy(&write\_permission);

    sem\_destroy(&read\_permission);

    return 0;

}

Output : -

