**30. Write C programs to demonstrate the following thread related concepts.**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#define NUM\_THREADS 3

// Shared data and synchronization primitives

pthread\_mutex\_t lock;

pthread\_cond\_t cond\_var;

int shared\_data = 0; // Shared resource

int ready = 0; // Condition flag

// Function executed by producer thread

void \*producer(void \*arg) {

pthread\_mutex\_lock(&lock);

printf("Producer: Preparing data...\n");

sleep(2); // Simulate data preparation

shared\_data = 42; // Set shared data

ready = 1; // Signal that data is ready

printf("Producer: Data is ready (shared\_data = %d).\n", shared\_data);

pthread\_cond\_signal(&cond\_var); // Signal the consumer

pthread\_mutex\_unlock(&lock);

return NULL;

}

// Function executed by consumer threads

void \*consumer(void \*arg) {

int thread\_id = \*(int \*)arg;

pthread\_mutex\_lock(&lock);

while (!ready) {

printf("Consumer %d: Waiting for data...\n", thread\_id);

pthread\_cond\_wait(&cond\_var, &lock); // Wait for producer to signal

}

printf("Consumer %d: Consumed data (shared\_data = %d).\n", thread\_id, shared\_data);

pthread\_mutex\_unlock(&lock);

return NULL;

}

int main() {

pthread\_t threads[NUM\_THREADS];

int thread\_ids[NUM\_THREADS];

int i;

// Initialize mutex and condition variable

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

pthread\_cond\_init(&cond\_var, NULL);

// Create the producer thread

pthread\_create(&threads[0], NULL, producer, NULL);

// Create the consumer threads

for (i = 1; i < NUM\_THREADS; i++) {

thread\_ids[i] = i;

pthread\_create(&threads[i], NULL, consumer, &thread\_ids[i]);

}

// Wait for all threads to complete

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

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

}

// Clean up

pthread\_mutex\_destroy(&lock);

pthread\_cond\_destroy(&cond\_var);

printf("Main: All threads have completed.\n");

return 0;

}

Output:

