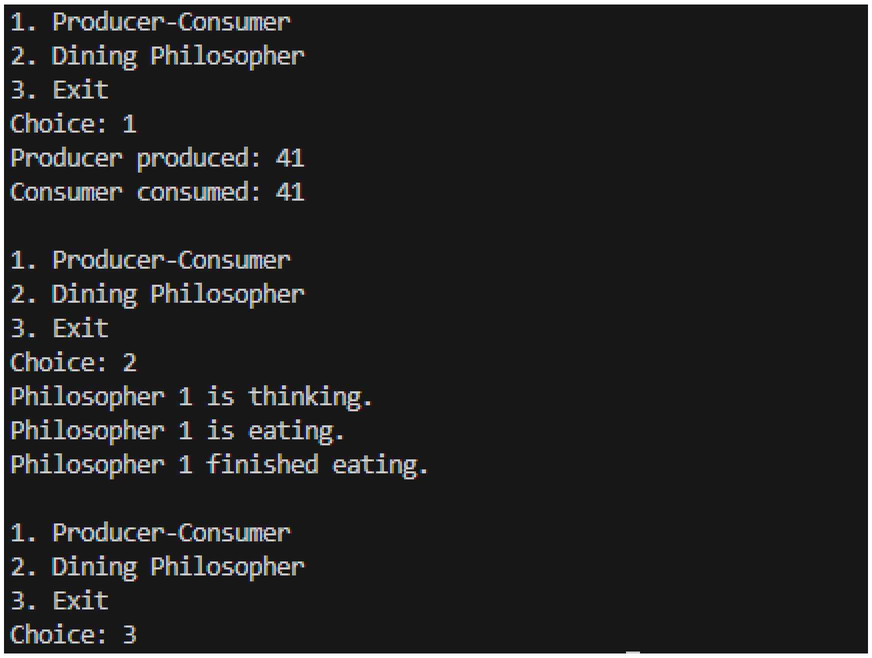
Write a C program to simulate:
a) Producer-Consumer problem using semaphores.
b) Dining-Philosopher's problem

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

// Producer-Consumer
int buffer;
sem_t empty, full;
pthread_mutex_t mutex;

void* producer(void* arg) {
    int item = rand() % 100;
    sem_wait(&empty);
    pthread_mutex_lock(&mutex);

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

    pthread_mutex_unlock(&mutex);
    sem_post(&full);
    return NULL;
}

void* consumer(void* arg) {
    int item;
    sem_wait(&full);
    pthread_mutex_lock(&mutex);

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

    pthread_mutex_unlock(&mutex);
    sem_post(&empty);
    return NULL;
}

void run_producer_consumer() {
    pthread_t p, c;
    sem_init(&empty, 0, 1);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);

    pthread_create(&p, NULL, producer, NULL);
    pthread_create(&c, NULL, consumer, NULL);

    pthread_join(p, NULL);
    pthread_join(c, NULL);

    sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);
}

// Dining Philosophers
sem_t forks[2];

void* philosopher(void* arg) {
    int id = *(int*)arg;

    printf("Philosopher %d is thinking.\n", id);
    sleep(1);

    sem_wait(&forks[0]);
    sem_wait(&forks[1]);

    printf("Philosopher %d is eating.\n", id);
    sleep(1);

    sem_post(&forks[0]);
    sem_post(&forks[1]);
    printf("Philosopher %d finished eating.\n", id);
    return NULL;
}

void run_dining_philosophers() {
    pthread_t phil;
    int id = 1;

    sem_init(&forks[0], 0, 1);
    sem_init(&forks[1], 0, 1);

    pthread_create(&phil, NULL, philosopher, &id);
    pthread_join(phil, NULL);

    sem_destroy(&forks[0]);
    sem_destroy(&forks[1]);
}

int main() {
    int choice;
    while (1) {
        printf("\n1. Producer-Consumer\n2. Dining Philosopher\n3. Exit\nChoice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: run_producer_consumer(); break;
            case 2: run_dining_philosophers(); break;
            case 3: exit(0);
            default: printf("Invalid choice!\n");
        }
    }
}
```

OUTPUT



```
1. Producer-Consumer
2. Dining Philosopher
3. Exit
Choice: 1
Producer produced: 41
Consumer consumed: 41

1. Producer-Consumer
2. Dining Philosopher
3. Exit
Choice: 2
Philosopher 1 is thinking.
Philosopher 1 is eating.
Philosopher 1 finished eating.

1. Producer-Consumer
2. Dining Philosopher
3. Exit
Choice: 3
```