Operating Systems Lab 8 | Assignment 8
Arnav Samal – 122CS0107

Q1.

1.Producer consumer problem with a shared buffer of size 5

- Main thread creates 3 producer threads and 3 consumer threads
- Each producer thread produces 4 items into the shared buffer
- Each consumer thread consumes 4 items from the shared buffer
- After a producer/consumer thread produces/consumes one item, it sleeps for 1s

Code:

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

#define BUFFER_SIZE 5
#define MAX_ITEMS 4

sem_t mutex;
sem_t FreeBuffers;
sem_t LoadedBuffers;

int buffer[BUFFER_SIZE];
int in = 0;
int out = 0;

void* producer(void* arg) {
   int num = *(int *)arg;
   for (int i = 0; i < MAX_ITEMS; i++) {
      sem_wait(&FreeBuffers);
      sem_wait(&mutex);

         buffer[in] = num * 10 + i;
         printf("Producer %d produced %d\n", num+1, buffer[in]);
         in = (in + 1) % BUFFER_SIZE;

      sem_post(&mutex);
      sem_post(&LoadedBuffers);

      sleep(1);
   }
   return NULL;
}

void* consumer(void* arg) {
   int num = *(int *)arg;
   for (int i = 0; i < MAX_ITEMS; i++) {
```

```
        sem_wait(&LoadedBuffers);
        sem_wait(&mutex);

            int item = buffer[out];
            printf("Consumer %d consumed %d\n", num+1, item);
            out = (out + 1) % BUFFER_SIZE;

        sem_post(&mutex);
        sem_post(&FreeBuffers);

        sleep(1);
    }
    return NULL;
}

int main() {
    sem_init(&mutex, 0, 1);
    sem_init(&FreeBuffers, 0, BUFFER_SIZE);
    sem_init(&LoadedBuffers, 0, 0);

    pthread_t producers[3];
    pthread_t consumers[3];
    int producer_nums[3] = {0, 1, 2};
    int consumer_nums[3] = {0, 1, 2};

    // Creating Producer Threads
    for (int i = 0; i < 3; i++) {
        pthread_create(&producers[i], NULL, producer, &producer_nums[i]);
    }

    // Creating Consumer Threads
    for (int i = 0; i < 3; i++) {
        pthread_create(&consumers[i], NULL, consumer, &consumer_nums[i]);
    }

    // Waiting for Producer Threads to Finish
    for (int i = 0; i < 3; i++) {
        pthread_join(producers[i], NULL);
    }

    // Waiting for Consumer Threads to Finish
    for (int i = 0; i < 3; i++) {
        pthread_join(consumers[i], NULL);
    }

    // Destroy semaphores
    sem_destroy(&mutex);
    sem_destroy(&FreeBuffers);
    sem_destroy(&LoadedBuffers);

    return 0;
}
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OS/Lab_8
File  Edit  View  Search  Terminal  Help
Consumer 2 consumed 23
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ gcc q1.c -lpthread
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ ./a.out
Producer 1 produced 0
Consumer 1 consumed 0
Producer 3 produced 20
Producer 2 produced 10
Consumer 2 consumed 20
Consumer 3 consumed 10
Producer 1 produced 1
Consumer 1 consumed 1
Producer 3 produced 21
Producer 2 produced 11
Consumer 2 consumed 21
Consumer 3 consumed 11
Producer 1 produced 2
Consumer 1 consumed 2
Producer 3 produced 22
Producer 2 produced 12
Consumer 2 consumed 22
Consumer 3 consumed 12
Producer 1 produced 3
Producer 3 produced 23
Consumer 3 consumed 3
Consumer 2 consumed 23
Producer 2 produced 13
Consumer 1 consumed 13
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ touch q2.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ gcc q2.c -lpthread
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ ./a.out
```

Q2.

2.Dining philosophers' problem with 5 philosophers and 5 chopsticks

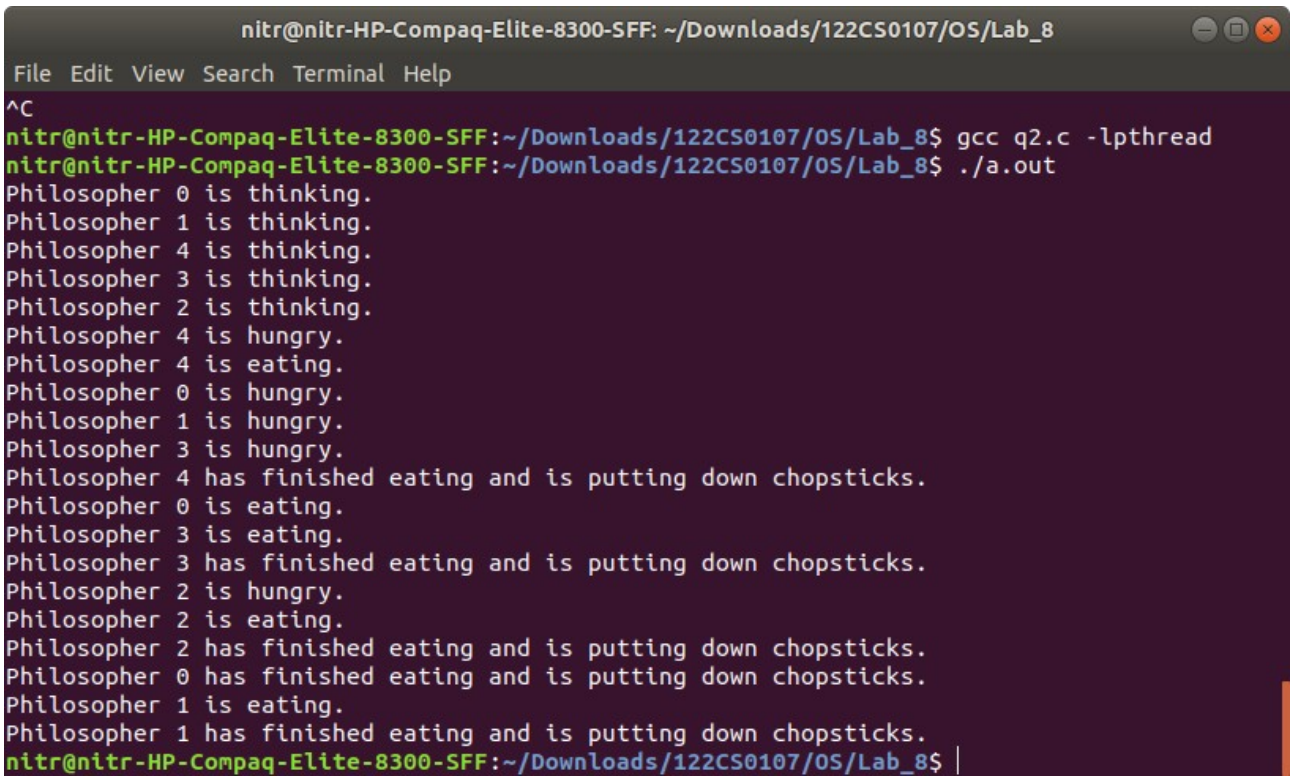
- And odd philosopher picks up her/his left chopstick first then the right chopstick
- And even the philosopher picks up her/his right chopstick first then the left chopstick

Code:

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

#define NUM_PHILOSOPHERS 5

sem_t chopsticks[NUM_PHILOSOPHERS];

void* philosopher(void* num) {
    int id = *(int*)num;
    //while (1) {
        printf("Philosopher %d is thinking.\n", id);
        sleep(rand() % 3);

        printf("Philosopher %d is hungry.\n", id);

        if (id % 2 == 0) {
            sem_wait(&chopsticks[(id + 1) % NUM_PHILOSOPHERS]);
            sem_wait(&chopsticks[id]);
```

```
        } else {
            sem_wait(&chopsticks[id]);
            sem_wait(&chopsticks[(id + 1) % NUM_PHILOSOPHERS]);
        }

        printf("Philosopher %d is eating.\n", id);
        sleep(rand() % 3);

        sem_post(&chopsticks[id]);
        sem_post(&chopsticks[(id + 1) % NUM_PHILOSOPHERS]);

        printf("Philosopher %d has finished eating and is putting down chopsticks.\n", id);
    //}

    return NULL;
}

int main() {
    pthread_t philosophers[NUM_PHILOSOPHERS];
    int philosopher_ids[NUM_PHILOSOPHERS];

    // Initialize semaphores
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        sem_init(&chopsticks[i], 0, 1);
        philosopher_ids[i] = i;
    }

    // Create philosopher threads
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_create(&philosophers[i], NULL, philosopher, &philosopher_ids[i]);
    }

    // Join philosopher threads
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        pthread_join(philosophers[i], NULL);
    }

    // Cleanup
    for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
        sem_destroy(&chopsticks[i]);
    }

    return 0;
}
```

Output:


nitr@nitr-HP-Compaq-Elite-8300-SFF: ~/Downloads/122CS0107/OS/Lab_8
File Edit View Search Terminal Help
^C
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ gcc q2.c -lpthread
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$ ./a.out
Philosopher 0 is thinking.
Philosopher 1 is thinking.
Philosopher 4 is thinking.
Philosopher 3 is thinking.
Philosopher 2 is thinking.
Philosopher 4 is hungry.
Philosopher 4 is eating.
Philosopher 0 is hungry.
Philosopher 1 is hungry.
Philosopher 3 is hungry.
Philosopher 4 has finished eating and is putting down chopsticks.
Philosopher 0 is eating.
Philosopher 3 is eating.
Philosopher 3 has finished eating and is putting down chopsticks.
Philosopher 2 is hungry.
Philosopher 2 is eating.
Philosopher 2 has finished eating and is putting down chopsticks.
Philosopher 0 has finished eating and is putting down chopsticks.
Philosopher 1 is eating.
Philosopher 1 has finished eating and is putting down chopsticks.
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_8$