NAME: KUMARI SHUBHANGINI
REGI NO.: 2241019442
SECTION: 31

DOS LAB ASSIGNMENT- 5: Implementation of synchronization using semaphore

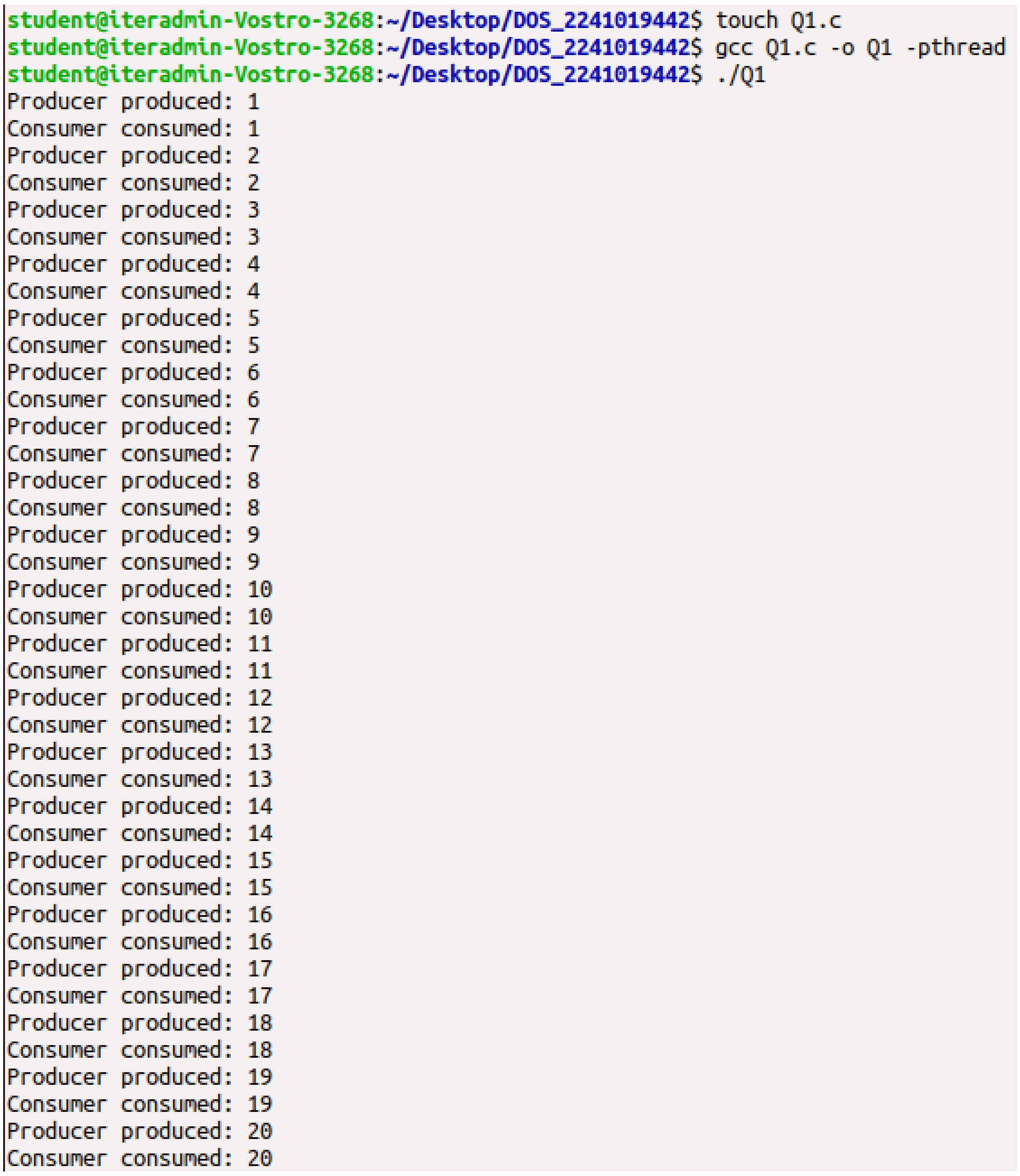
Q1)Producer-Consumer problem

Write a C program to implement the producer-consumer program.
Where: * Producer generates integers from 1 to 100. * Consumer processes the numbers.

CODE:

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define BUFFER_SIZE 10
#define ITERATIONS 20
int buffer[BUFFER_SIZE];
int in = 0, out = 0;
sem_t empty;
sem_t full;
pthread_mutex_t mutex;
void *producer(void *arg) {
    for (int i = 1; i <= ITERATIONS; i++) {
        int item = i;
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
        printf("Producer produced: %d\n", item);
        in = (in + 1) % BUFFER_SIZE;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
        sleep(1);     }
    return NULL;}
void *consumer(void *arg) {
    for (int i = 1; i <= ITERATIONS; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumer consumed: %d\n", item);
        out = (out + 1) % BUFFER_SIZE;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
        sleep(1);    }
    return NULL;}

int main() {
    pthread_t prod_thread, cons_thread;
    sem_init(&empty, 0, BUFFER_SIZE);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);

    pthread_create(&prod_thread, NULL, producer, NULL);
    pthread_create(&cons_thread, NULL, consumer, NULL);
    pthread_join(prod_thread, NULL);
    pthread_join(cons_thread, NULL);
    sem_destroy(&empty);
    sem_destroy(&full);
    pthread_mutex_destroy(&mutex);

    return 0;}
```

OUTPUT:



```
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ touch Q1.c
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ gcc Q1.c -o Q1 -pthread
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ ./Q1
Producer produced: 1
Consumer consumed: 1
Producer produced: 2
Consumer consumed: 2
Producer produced: 3
Consumer consumed: 3
Producer produced: 4
Consumer consumed: 4
Producer produced: 5
Consumer consumed: 5
Producer produced: 6
Consumer consumed: 6
Producer produced: 7
Consumer consumed: 7
Producer produced: 8
Consumer consumed: 8
Producer produced: 9
Consumer consumed: 9
Producer produced: 10
Consumer consumed: 10
Producer produced: 11
Consumer consumed: 11
Producer produced: 12
Consumer consumed: 12
Producer produced: 13
Consumer consumed: 13
Producer produced: 14
Consumer consumed: 14
Producer produced: 15
Consumer consumed: 15
Producer produced: 16
Consumer consumed: 16
Producer produced: 17
Consumer consumed: 17
Producer produced: 18
Consumer consumed: 18
Producer produced: 19
Consumer consumed: 19
Producer produced: 20
Consumer consumed: 20
```

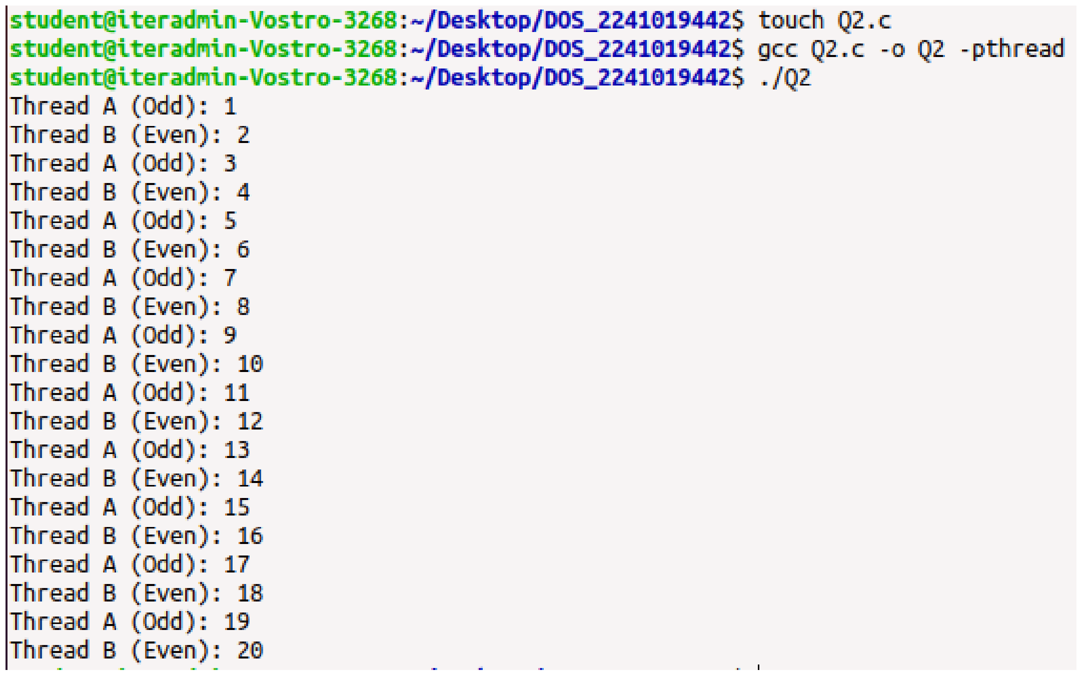
Q2) Alternating Numbers with Two Threads

Write a program to print 1, 2, 3 … upto 20.
Create threads where two threads print numbers alternately.
Thread A prints odd numbers: 1, 3, 5 ...
Thread B prints even numbers: 2, 4, 6 …

CODE:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define MAX_NUMBER 20
sem_t sem_odd, sem_even;
void *print_odd(void *arg) {
    for (int i = 1; i <= MAX_NUMBER; i += 2) {
        sem_wait(&sem_odd);
        printf("Thread A (Odd): %d\n", i);
        sem_post(&sem_even);
    }
    return NULL;
}
void *print_even(void *arg) {
    for (int i = 2; i <= MAX_NUMBER; i += 2) {
        sem_wait(&sem_even);
        printf("Thread B (Even): %d\n", i);
        sem_post(&sem_odd);
    }
    return NULL;}

int main() {
    pthread_t thread_odd, thread_even;
    sem_init(&sem_odd, 0, 1);
    sem_init(&sem_even, 0, 0);
    pthread_create(&thread_odd, NULL, print_odd, NULL);
    pthread_create(&thread_even, NULL, print_even, NULL);
    pthread_join(thread_odd, NULL);
    pthread_join(thread_even, NULL);
    sem_destroy(&sem_odd);
    sem_destroy(&sem_even);
    return 0;
}
```

OUTPUT:



```
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ touch Q2.c
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ gcc Q2.c -o Q2 -pthread
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ ./Q2
Thread A (Odd): 1
Thread B (Even): 2
Thread A (Odd): 3
Thread B (Even): 4
Thread A (Odd): 5
Thread B (Even): 6
Thread A (Odd): 7
Thread B (Even): 8
Thread A (Odd): 9
Thread B (Even): 10
Thread A (Odd): 11
Thread B (Even): 12
Thread A (Odd): 13
Thread B (Even): 14
Thread A (Odd): 15
Thread B (Even): 16
Thread A (Odd): 17
Thread B (Even): 18
Thread A (Odd): 19
Thread B (Even): 20
```

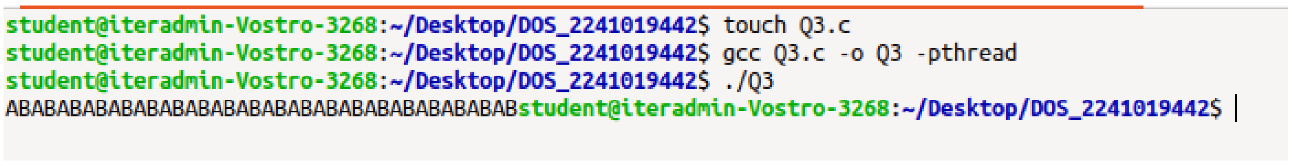
Q3) Alternating Characters

Write a program to create two threads that print characters (A and B) alternately such as ABABABABA…. upto 20. Use semaphores to synchronize the threads.
Thread A prints A.
Thread B prints B.

CODE:

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

#define MAX_COUNT 20

sem_t sem_a, sem_b;
void *print_a(void *arg) {
    for (int i = 0; i < MAX_COUNT; i++) {
        sem_wait(&sem_a);
        printf("A");
        fflush(stdout);
        sem_post(&sem_b);
    }
    return NULL;
}
void *print_b(void *arg) {
    for (int i = 0; i < MAX_COUNT; i++) {
        sem_wait(&sem_b);
        printf("B");
        fflush(stdout);
        sem_post(&sem_a);
    }
    return NULL;}

int main() {
    pthread_t thread_a, thread_b;
    sem_init(&sem_a, 0, 1);
    sem_init(&sem_b, 0, 0);
    pthread_create(&thread_a, NULL, print_a, NULL);
    pthread_create(&thread_b, NULL, print_b, NULL);
    pthread_join(thread_a, NULL);
    pthread_join(thread_b, NULL);
    sem_destroy(&sem_a);
    sem_destroy(&sem_b);

    return 0;}
```

OUTPUT:



```
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ touch Q3.c
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ gcc Q3.c -o Q3 -pthread
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ ./Q3
ABABABABABABABABABABABABABABABABABABABABstudent@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ |
```

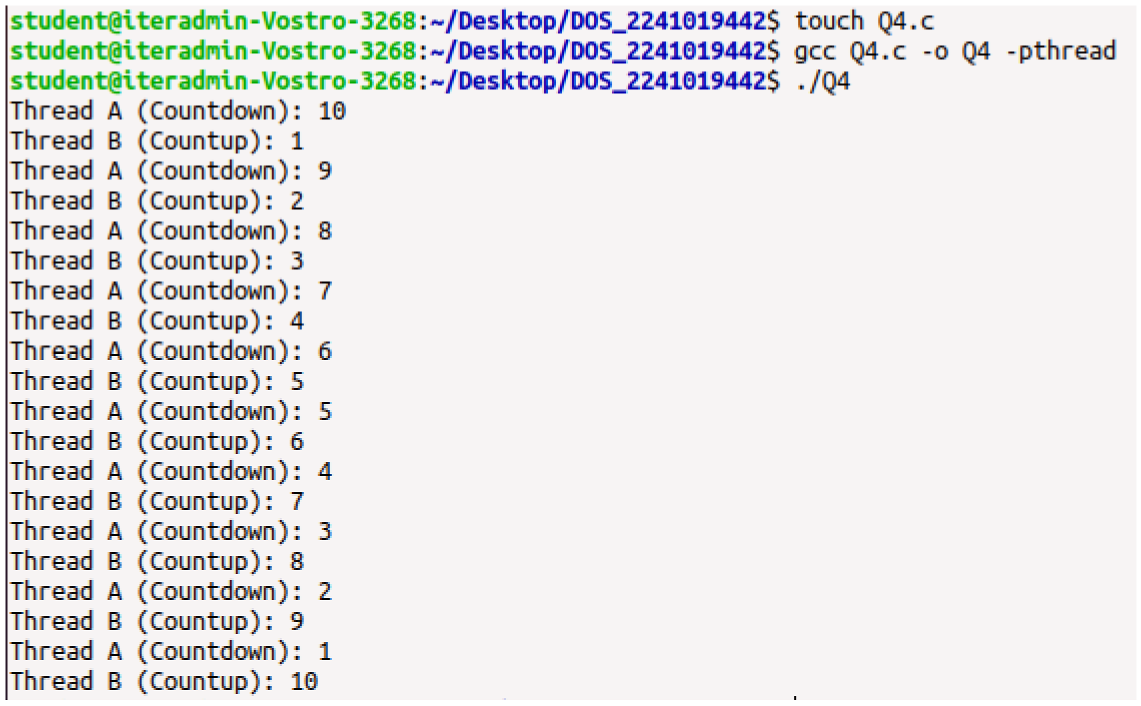
Q4) Countdown and Count up Problem

Write a program create two threads where:
Thread A counts down from 10 to 1.
Thread B counts up from 1 to 10.

CODE:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define MAX_COUNT 10
sem_t sem_a, sem_b;
void *countdown(void *arg) {
    for (int i = MAX_COUNT; i >= 1; i--) {
        sem_wait(&sem_a);
        printf("Thread A (Countdown): %d\n", i);
        sem_post(&sem_b);
    }
    return NULL;
}
void *countup(void *arg) {
    for (int i = 1; i <= MAX_COUNT; i++) {
        sem_wait(&sem_b);
        printf("Thread B (Countup): %d\n", i);
        sem_post(&sem_a);
    }
    return NULL;
}
int main() {
    pthread_t thread_a, thread_b;
    sem_init(&sem_a, 0, 1);
    sem_init(&sem_b, 0, 0);
    pthread_create(&thread_a, NULL, countdown, NULL);
    pthread_create(&thread_b, NULL, countup, NULL);
    pthread_join(thread_a, NULL);
    pthread_join(thread_b, NULL);
    sem_destroy(&sem_a);
    sem_destroy(&sem_b); return 0;}
```

OUTPUT:



```
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ touch Q4.c
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ gcc Q4.c -o Q4 -pthread
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ ./Q4
Thread A (Countdown): 10
Thread B (Countup): 1
Thread A (Countdown): 9
Thread B (Countup): 2
Thread A (Countdown): 8
Thread B (Countup): 3
Thread A (Countdown): 7
Thread B (Countup): 4
Thread A (Countdown): 6
Thread B (Countup): 5
Thread A (Countdown): 5
Thread B (Countup): 6
Thread A (Countdown): 4
Thread B (Countup): 7
Thread A (Countdown): 3
Thread B (Countup): 8
Thread A (Countdown): 2
Thread B (Countup): 9
Thread A (Countdown): 1
Thread B (Countup): 10
```

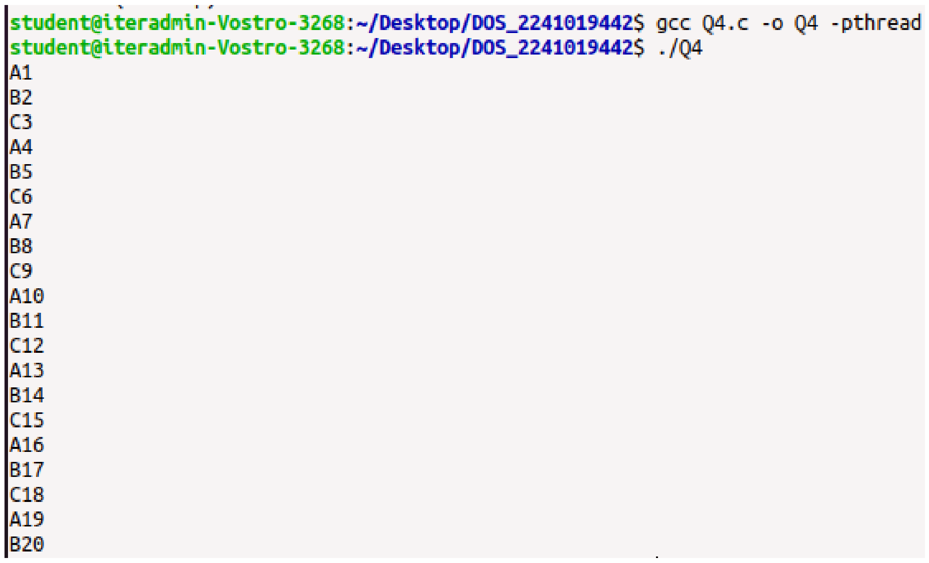
Q5) Sequence Printing using Threads Problem

Write a program that creates three threads: Thread A, Thread B, and Thread C. The threads must print numbers in the following sequence: A1, B2, C3, A4, B5, C6 … upto 20 numbers.
Thread A prints A1, A4, A7, …
Thread B prints B2, B5, B8, …
Thread C prints C3, C6, C9, …

CODE:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#define MAX_COUNT 20
sem_t sem_a, sem_b, sem_c;
void *print_a(void *arg) {
    for (int i = 1; i <= MAX_COUNT; i += 3) {
        sem_wait(&sem_a);
        printf("A%d\n", i);
        sem_post(&sem_b);
    }
    return NULL;}
void *print_b(void *arg) {
    for (int i = 2; i <= MAX_COUNT; i += 3) {
        sem_wait(&sem_b);
        printf("B%d\n", i);
        sem_post(&sem_c);     }
    return NULL;}
void *print_c(void *arg) {
    for (int i = 3; i <= MAX_COUNT; i += 3) {
        sem_wait(&sem_c);
        printf("C%d\n", i);
        sem_post(&sem_a);
    }
    return NULL;}

int main() {
    pthread_t thread_a, thread_b, thread_c;
    sem_init(&sem_a, 0, 1);
    sem_init(&sem_b, 0, 0);
    sem_init(&sem_c, 0, 0);
    pthread_create(&thread_a, NULL, print_a, NULL);
    pthread_create(&thread_b, NULL, print_b, NULL);
    pthread_create(&thread_c, NULL, print_c, NULL);
    pthread_join(thread_a, NULL);
    pthread_join(thread_b, NULL);
    pthread_join(thread_c, NULL);
    sem_destroy(&sem_a);
    sem_destroy(&sem_b);
    sem_destroy(&sem_c);
    return 0;
}
```

OUTPUT:



```
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ gcc Q4.c -o Q4 -pthread
student@iteradmin-Vostro-3268:~/Desktop/DOS_2241019442$ ./Q4
A1
B2
C3
A4
B5
C6
A7
B8
C9
A10
B11
C12
A13
B14
C15
A16
B17
C18
A19
B20
```