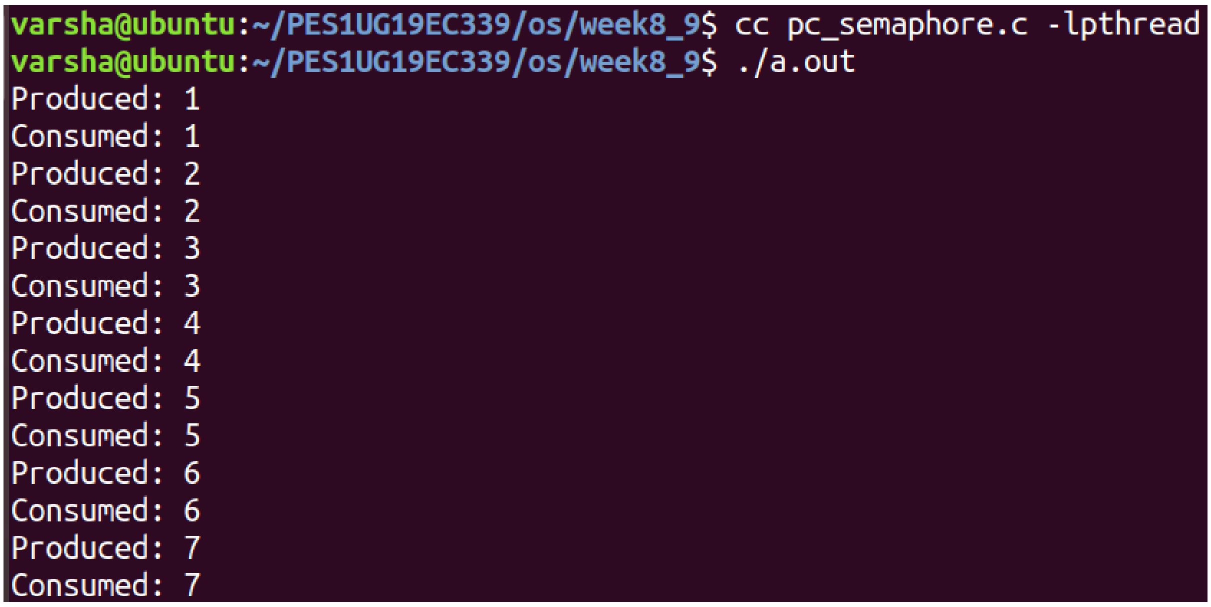
NAME: VARSHA D KULKARNI SRN: PES1UG19EC339

OS-LAB-WEEK-8-9

PROGRAM 1: Write a C program to implement Producer Consumer problem using Semaphores

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

#define BUF_SIZE 50

int front = 0, rear = 0;
int *buffer;

sem_t mutex;
sem_t empty;
sem_t full;

void *producer();
void *consumer();

int main()
{
   buffer = (int *)malloc(sizeof(int) * BUF_SIZE);
   pthread_t thread1, thread2;

   sem_init(&mutex, 0, 1);
   sem_init(&empty, 0, 1);
   sem_init(&full, 0, 0);
   pthread_create(&thread1, NULL, producer, NULL);
   sleep(1);

   pthread_create(&thread2, NULL, consumer, NULL);
   pthread_join(thread1, NULL);
   pthread_join(thread2, NULL);

   free(buffer);
   return 0;
}

void *consumer()
{
   while (true)
   {
        sem_wait(&full);
        sem_wait(&mutex);

        int consumed = buffer[rear];
        printf("Consumed: %d\n", consumed);
        sleep(1);

        rear = (rear + 1) % BUF_SIZE;
        sem_post(&mutex);
        sem_post(&empty);
    }
}

void *producer()
{
    int item = 0;
    while (true)
    {
        sem_wait(&empty);
        sem_wait(&mutex);

        item++;
        printf("Produced: %d\n", item);
        buffer[front] = item;

        sem_post(&mutex);
        sem_post(&full);
        front = (front + 1) % BUF_SIZE;
    }
}
```

OUTPUT:

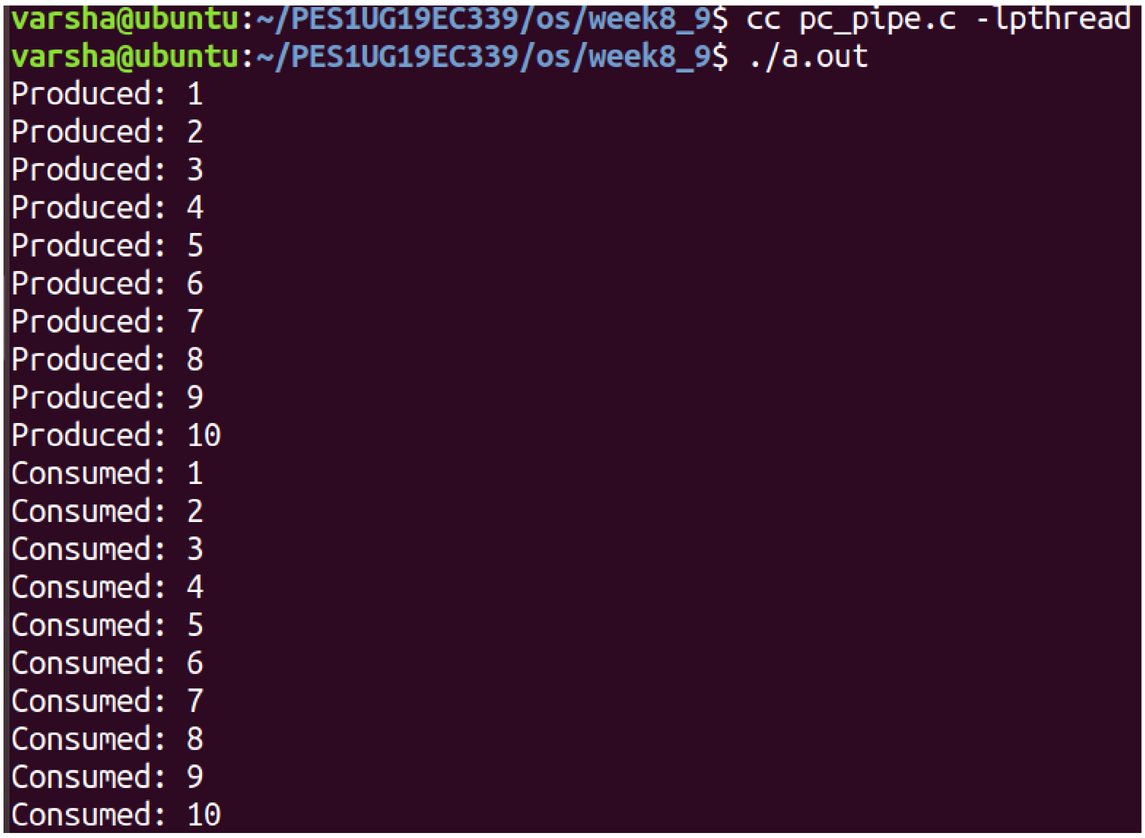


```
varsha@ubuntu:~/PES1UG19EC339/os/week8_9$ cc pc_semaphore.c -lpthread
varsha@ubuntu:~/PES1UG19EC339/os/week8_9$ ./a.out
Produced: 1
Consumed: 1
Produced: 2
Consumed: 2
Produced: 3
Consumed: 3
Produced: 4
Consumed: 4
Produced: 5
Consumed: 5
Produced: 6
Consumed: 6
Produced: 7
Consumed: 7
```

PROGRAM 2: Write a C program to implement Producer Consumer problem using Pipes

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <wait.h>

void producer(FILE *);
void consumer(FILE *);

int main()
{
   int file_descriptor[2];
   if (pipe(file_descriptor) < 0)
      exit(1);

   FILE *pipe_read = fdopen(file_descriptor[0], "r");
   FILE *pipe_write = fdopen(file_descriptor[1], "w");

   pid_t producer_pid = fork();
   if (producer_pid == 0)
   {
      fclose(pipe_read);
      producer(pipe_write);
   }

   pid_t consumer_pid = fork();
   if (consumer_pid == 0)
   {
      fclose(pipe_write);
      consumer(pipe_read);
   }

   fclose(pipe_read);
   fclose(pipe_write);
   wait(NULL);
   wait(NULL);

   return 0;
}

void producer(FILE *pipe_write)
{
   int item = 0;
   for (int i = 0; i < 10; ++i)
   {
      item++;
      fprintf(pipe_write, "%d ", item);
```

```
        printf("Produced: %d\n", item);
    }
    fclose(pipe_write);
    exit(0);
}

void consumer(FILE *pipe_read)
{
    int consumed, n;
    while (true)
    {
        n = fscanf(pipe_read, "%d", &consumed);
        if (n == 1)
            printf("Consumed: %d\n", consumed);
        else
            break;
    }
    fclose(pipe_read);
    exit(0);
}
```



```
varsha@ubuntu:~/PES1UG19EC339/os/week8_9$ cc pc_pipe.c -lpthread
varsha@ubuntu:~/PES1UG19EC339/os/week8_9$ ./a.out
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Produced: 10
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Consumed: 5
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10
```