[CSS553]

Name: Rahul Ranjan
Roll No: 20CS8016

Assignment 5

1. Write program in C where the parent thread creates 10 threads to find prime number. The child threads increments the count of primes in a globally declared integer variable `count`. The acess to `count` should be protected by semaphores. Solve the problem using POSIX unnamed semaphores.

2. Solve the above problem 1 by creating multiple process instead of threads and implement access synchronization using named semaphores.

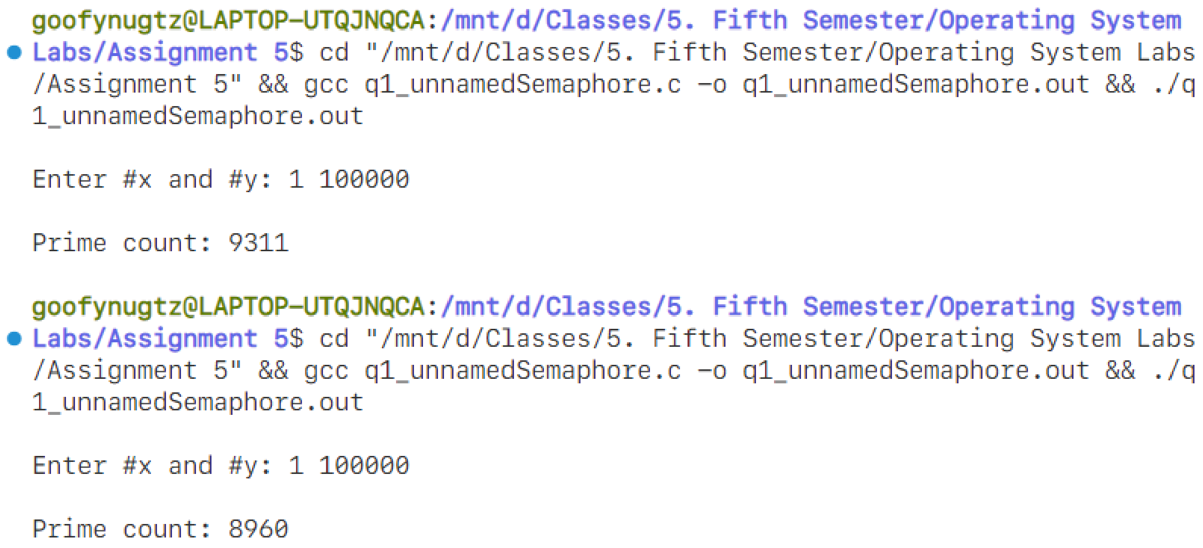
3. Implement a solution for the producer-consumer (infinite buffer) problem using semaphores.

Answer 1 – Primes using threads with unnamed semaphores.

```c
#include <stdio.h>
#include <stdbool.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#define TC 10

pthread_t *c_thread;
int count = 0;
sem_t semaphore;

typedef struct c_args {
  int x, y;
} c_args;

bool isPrime(int n){
  if (n <= 1)
    return false;
  for (int i = 2; i*i <= n; i++)
    if (n % i == 0) return false;
  return true;
}

void * getPrimes(void * a){
  c_args * args = (c_args*)a;
  int start = (int)((c_args*)args)->x;
  int end = (int)((c_args*)args)->y;
  for (int i = start; i <= end; i++)
    if(isPrime(i)){
      sem_wait(&semaphore);
      count++;
      sem_post(&semaphore);
    }
  free(a);
  return 0;
}

```

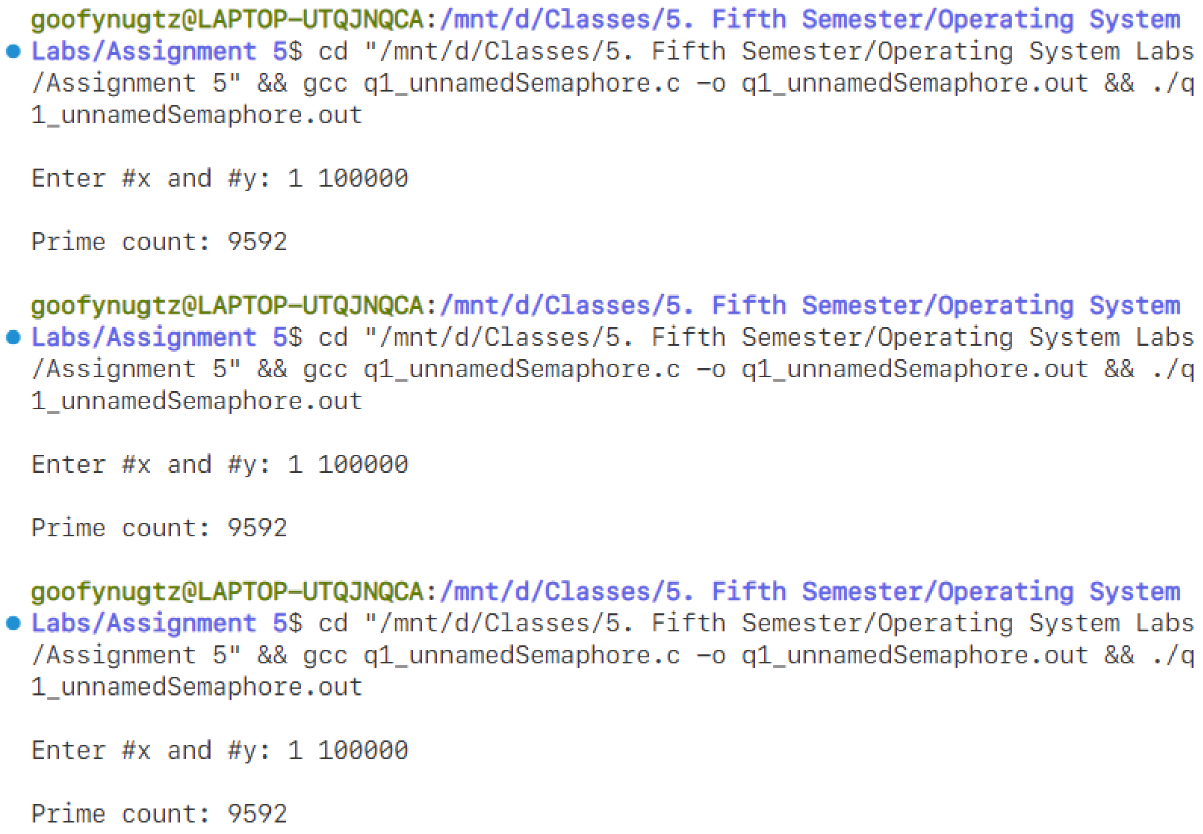
```c
int main(void){

  sem_init(&semaphore, 0, 1);
  printf("\nEnter #x and #y: ");
  int x,y;
  scanf("%d%d", &x, &y);
  printf("\n");
  int cc = TC;
  c_thread = (pthread_t*) malloc(sizeof(pthread_t)*cc);
  c_args **args = (c_args**) malloc(sizeof(c_args*)*cc);
  for (int i = 0; i < cc; i++){
    args[i] = (c_args*) malloc(sizeof(c_args));
    args[i]->x = i*(y-x)/cc + x+1;
    args[i]->y = (i+1)*(y-x)/cc + x;
    pthread_create(&c_thread[i], NULL, getPrimes, (void*)(args[i]));
  }
  free(args);
  int c;
  for (int i = 0; i < TC; i++)
    c = pthread_join(c_thread[i], NULL);
  sem_destroy(&semaphore);
  printf("Prime count: %d\n\n", count);

  return 0;
}
```

>> Race condition occuring when `sem_wait()` and `sem_post()` was commented out.



```
  goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
● Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
  /Assignment 5" && gcc q1_unnamedSemaphore.c -o q1_unnamedSemaphore.out && ./q
  1_unnamedSemaphore.out

  Enter #x and #y: 1 100000

  Prime count: 9311

  goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
● Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
  /Assignment 5" && gcc q1_unnamedSemaphore.c -o q1_unnamedSemaphore.out && ./q
  1_unnamedSemaphore.out

  Enter #x and #y: 1 100000

  Prime count: 8960
```

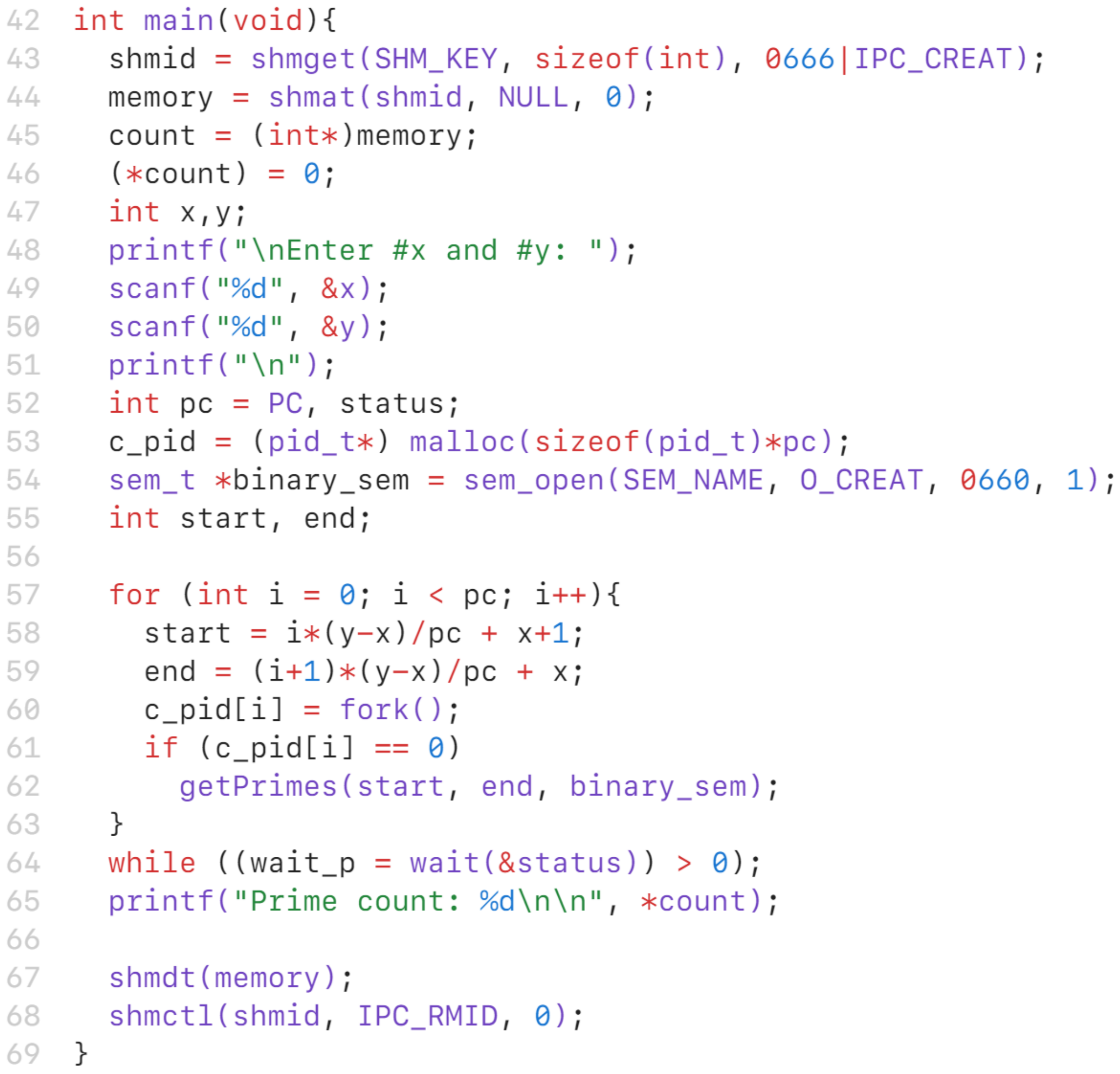
>> Output – Multiple runs



```
  goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
● Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
  /Assignment 5" && gcc q1_unnamedSemaphore.c -o q1_unnamedSemaphore.out && ./q
  1_unnamedSemaphore.out

  Enter #x and #y: 1 100000

  Prime count: 9592

  goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
● Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
  /Assignment 5" && gcc q1_unnamedSemaphore.c -o q1_unnamedSemaphore.out && ./q
  1_unnamedSemaphore.out

  Enter #x and #y: 1 100000

  Prime count: 9592

  goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
● Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
  /Assignment 5" && gcc q1_unnamedSemaphore.c -o q1_unnamedSemaphore.out && ./q
  1_unnamedSemaphore.out

  Enter #x and #y: 1 100000

  Prime count: 9592
```

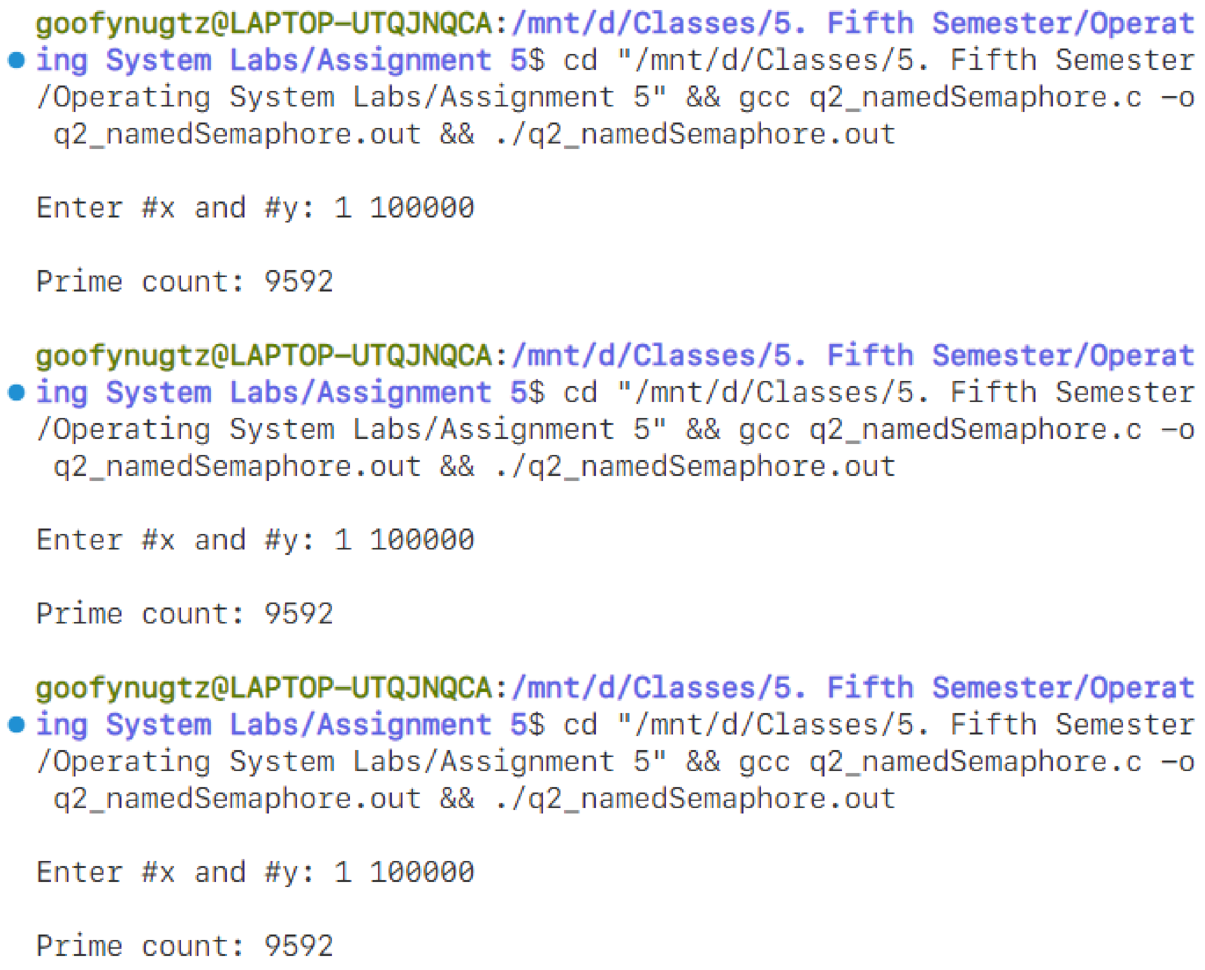
Answer 2 – Primes using processes with named semaphores.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdbool.h>
#include <fcntl.h>
#include <semaphore.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#define SEM_NAME "/mutex"
#define SHM_KEY 0x1234
#define PC 10

pid_t *c_pid, wait_p;
int shmid, *count;
void *memory;

bool isPrime(int n){
  if (n <= 1)
    return false;
  for (int i = 2; i*i <= n; i++)
    if (n % i == 0) return false;
  return true;
}

void getPrimes(int x, int y, sem_t *b_sem){
  for (int i = x; i <= y; i++)
    if(isPrime(i)){
      sem_wait(b_sem);
      void *c_m = shmat(shmid, NULL, 0);
      int *_count = (int*)c_m;
      (*_count) += 1;
      shmdt(c_m);
      sem_post(b_sem);
    }
  exit(EXIT_SUCCESS);
}

```



```
int main(void){
  shmid = shmget(SHM_KEY, sizeof(int), 0666|IPC_CREAT);
  memory = shmat(shmid, NULL, 0);
  count = (int*)memory;
  (*count) = 0;
  int x,y;
  printf("\nEnter #x and #y: ");
  scanf("%d", &x);
  scanf("%d", &y);
  printf("\n");
  int pc = PC, status;
  c_pid = (pid_t*) malloc(sizeof(pid_t)*pc);
  sem_t *binary_sem = sem_open(SEM_NAME, O_CREAT, 0660, 1);
  int start, end;

  for (int i = 0; i < pc; i++){
    start = i*(y-x)/pc + x+1;
    end = (i+1)*(y-x)/pc + x;
    c_pid[i] = fork();
    if (c_pid[i] == 0)
      getPrimes(start, end, binary_sem);
  }
  while ((wait_p = wait(&status)) > 0);
  printf("Prime count: %d\n\n", *count);

  shmdt(memory);
  shmctl(shmid, IPC_RMID, 0);
}
```

>> Output – Multiple runs



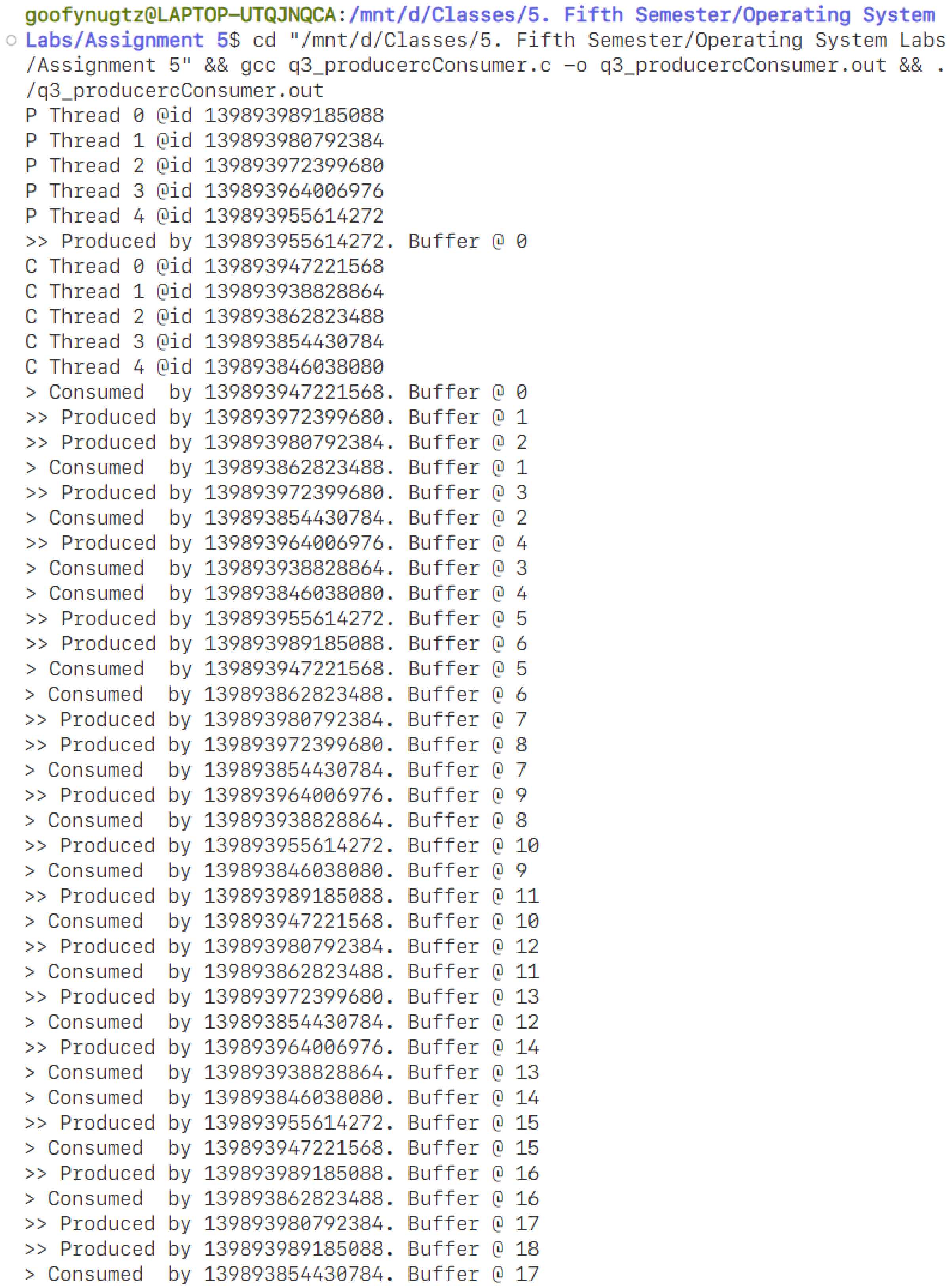
```
goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operat
ing System Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester
/Operating System Labs/Assignment 5" && gcc q2_namedSemaphore.c -o
 q2_namedSemaphore.out && ./q2_namedSemaphore.out

Enter #x and #y: 1 100000

Prime count: 9592

goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operat
ing System Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester
/Operating System Labs/Assignment 5" && gcc q2_namedSemaphore.c -o
 q2_namedSemaphore.out && ./q2_namedSemaphore.out

Enter #x and #y: 1 100000

Prime count: 9592

goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operat
ing System Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester
/Operating System Labs/Assignment 5" && gcc q2_namedSemaphore.c -o
 q2_namedSemaphore.out && ./q2_namedSemaphore.out

Enter #x and #y: 1 100000

Prime count: 9592
```

Answer 3 – Producer Consumer

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <semaphore.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <stdbool.h>
#include <pthread.h>
#define N (int)1e6

sem_t nrFull, nrEmpty, mutexPd, mutexCn;
int buffer, in, out;
pthread_t *p_id;
pthread_t *c_id;

void* producer(){
  while(1){
    sem_wait(&mutexPd);
    sem_wait(&nrEmpty);
    printf(">> Produced by %lu. Buffer @ %d\n", pthread_self(), in);
    in = (in+1) % N;
    buffer++;
    sleep(1);
    sem_post(&nrFull);
    sem_post(&mutexPd);
    sleep(1);
  }
}

void *generateProducer(void *args){
  for (int i = 0; i < (*(int*)args); i++){
    pthread_create(&p_id[i], NULL, producer, NULL);
    printf("P Thread %d @id %ld\n", i, p_id[i]);
  }
}

void* consumer(){
  while(1){
    sem_wait(&mutexCn);
    sem_wait(&nrFull);
    printf("> Consumed  by %lu. Buffer @ %d\n", pthread_self(), out);
    out = (out+1) % N;
    buffer--;
    sleep(1);
    sem_post(&nrEmpty);
    sem_post(&mutexCn);
    sleep(1);
  }
}
```

```c

void *generateConsumers(void *args){
  for (int i = 0; i < (*(int*)args); i++){
    pthread_create(&c_id[i], NULL, consumer, NULL);
    printf("C Thread %d @id %ld\n", i, c_id[i]);
  }
}

int main(void){

  int p_count = 5, c_count = 5;
  // printf("Enter #Producer #Consumer: ");
  // scanf("%d %d", &p_count, &c_count);

  sem_init(&nrFull, 0, 0);
  sem_init(&nrEmpty, 0, N);
  sem_init(&mutexPd, 0, 1);
  sem_init(&mutexCn, 0, 1);

  p_id = (pthread_t*) malloc(sizeof(pthread_t)* p_count);
  c_id = (pthread_t*) malloc(sizeof(pthread_t)* c_count);

  pthread_t gen_p;
  pthread_create(&gen_p, NULL, generateProducer, &p_count);
  pthread_t gen_c;
  pthread_create(&gen_c, NULL, generateConsumers, &c_count);

  while(1);
  return 0;
}
```

>> Output



```
 goofynugtz@LAPTOP-UTQJNQCA:/mnt/d/Classes/5. Fifth Semester/Operating System
○ Labs/Assignment 5$ cd "/mnt/d/Classes/5. Fifth Semester/Operating System Labs
 /Assignment 5" && gcc q3_producercConsumer.c -o q3_producercConsumer.out && .
 /q3_producercConsumer.out
 P Thread 0 @id 139893989185088
 P Thread 1 @id 139893980792384
 P Thread 2 @id 139893972399680
 P Thread 3 @id 139893964006976
 P Thread 4 @id 139893955614272
 >> Produced by 139893955614272. Buffer @ 0
 C Thread 0 @id 139893947221568
 C Thread 1 @id 139893938828864
 C Thread 2 @id 139893862823488
 C Thread 3 @id 139893854430784
 C Thread 4 @id 139893846038080
 > Consumed  by 139893947221568. Buffer @ 0
 >> Produced by 139893972399680. Buffer @ 1
 >> Produced by 139893980792384. Buffer @ 2
 > Consumed  by 139893862823488. Buffer @ 1
 >> Produced by 139893972399680. Buffer @ 3
 > Consumed  by 139893854430784. Buffer @ 2
 >> Produced by 139893964006976. Buffer @ 4
 > Consumed  by 139893938828864. Buffer @ 3
 > Consumed  by 139893846038080. Buffer @ 4
 >> Produced by 139893955614272. Buffer @ 5
 >> Produced by 139893989185088. Buffer @ 6
 > Consumed  by 139893947221568. Buffer @ 5
 > Consumed  by 139893862823488. Buffer @ 6
 >> Produced by 139893980792384. Buffer @ 7
 >> Produced by 139893972399680. Buffer @ 8
 > Consumed  by 139893854430784. Buffer @ 7
 >> Produced by 139893964006976. Buffer @ 9
 > Consumed  by 139893938828864. Buffer @ 8
 >> Produced by 139893955614272. Buffer @ 10
 > Consumed  by 139893846038080. Buffer @ 9
 >> Produced by 139893989185088. Buffer @ 11
 > Consumed  by 139893947221568. Buffer @ 10
 >> Produced by 139893980792384. Buffer @ 12
 > Consumed  by 139893862823488. Buffer @ 11
 >> Produced by 139893972399680. Buffer @ 13
 > Consumed  by 139893854430784. Buffer @ 12
 >> Produced by 139893964006976. Buffer @ 14
 > Consumed  by 139893938828864. Buffer @ 13
 > Consumed  by 139893846038080. Buffer @ 14
 >> Produced by 139893955614272. Buffer @ 15
 > Consumed  by 139893947221568. Buffer @ 15
 >> Produced by 139893989185088. Buffer @ 16
 > Consumed  by 139893862823488. Buffer @ 16
 >> Produced by 139893980792384. Buffer @ 17
 >> Produced by 139893989185088. Buffer @ 18
 > Consumed  by 139893854430784. Buffer @ 17
```

All compiler code are uploaded here.