TASK 1:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/types.h>

#include<pthread.h>

#include<time.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int in = 0, out = 0, count = 0;

void \*producer()

{

int next\_produced = 1;

while (1)

{

while (count == BUFFER\_SIZE)

{

sleep(rand()%12);

}

buffer[in] = next\_produced;

in = (in + 1) % BUFFER\_SIZE;

count++;

printf("Producer: Inserted %d into the buffer\n", next\_produced);

next\_produced++;

sleep(1);

}

pthread\_exit(NULL);

}

void \*consumer()

{

int next\_consumed=1;

while (1)

{

while (count == 0)

{

sleep(rand()%7);

}

next\_consumed = buffer[out];

out = (out + 1) % BUFFER\_SIZE;

count--;

printf("Consumer: Removed %d from the buffer\n", next\_consumed);

sleep(1);

}

pthread\_exit(NULL);

}

int main()

{

srand(time(0));

printf("Producer Consumer in Critical Situation\n");

pthread\_t p1,c1;

pthread\_create(&p1,NULL,producer,NULL);

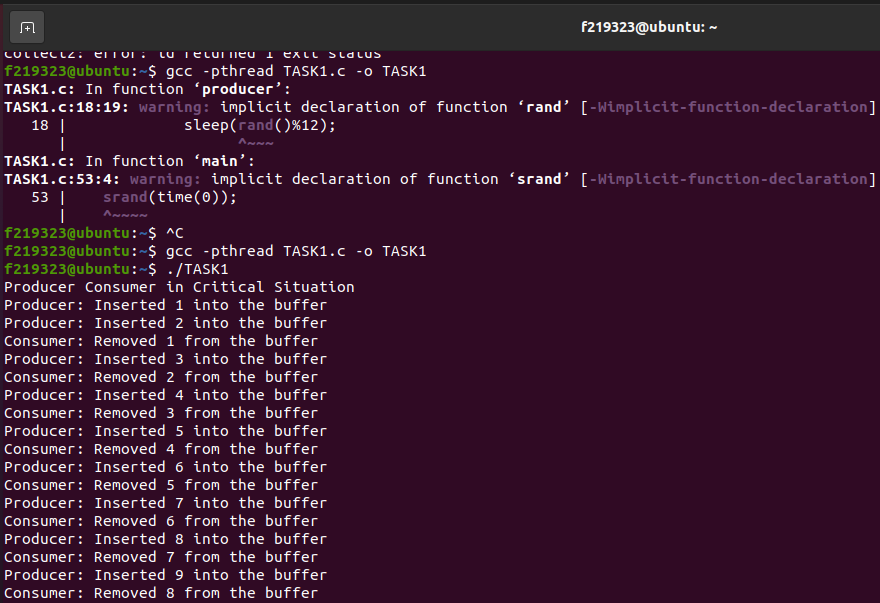
pthread\_create(&c1,NULL,consumer,NULL);

pthread\_join(p1,NULL);

pthread\_join(c1,NULL);

return 0;

}



TASK 2:  
#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

int counter = 0;

void \*increment(void \*arg)

{

int i;

for(i = 0; i < 100000; i++) {

counter++;

}

pthread\_exit(NULL);

}

int main()

{

pthread\_t threads[4];

int i;

for(i = 0; i < 4; i++) {

pthread\_create(&threads[i], NULL, increment, NULL);

}

for(i = 0; i < 4; i++) {

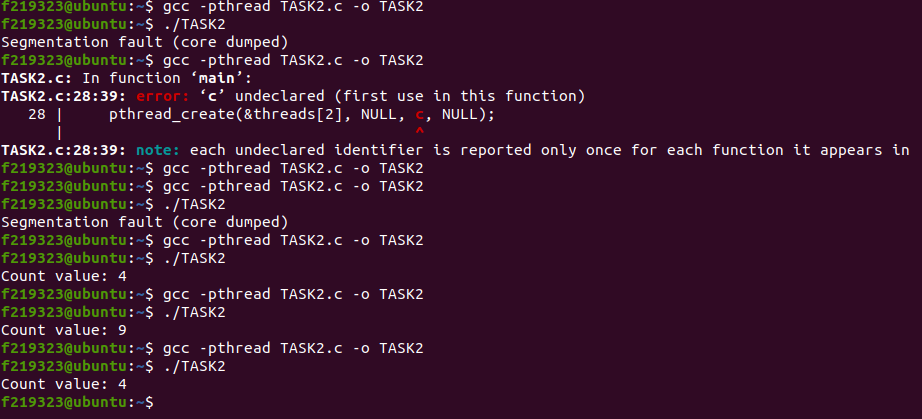
pthread\_join(threads[i], NULL);

}

printf("Final counter value: %d\n", counter);

return 0;

}



# TASK 3:

#include <stdio.h>

#include <stdbool.h>

#include <pthread.h>

#define NUM\_THREADS 2

bool flag[2] = {false, false};

int turn = 0;

void enter\_critical\_section(int thread\_num)

{

int other\_thread = 1 - thread\_num;

flag[thread\_num] = true;

turn = other\_thread;

while (flag[other\_thread] && turn == other\_thread);

}

void leave\_critical\_section(int thread\_num)

{

flag[thread\_num] = false;

}

void\* thread\_function(void\* arg)

{

int thread\_num = \*(int\*) arg;

for (int i = 0; i < 10; i++)

{

enter\_critical\_section(thread\_num);

printf("Thread %d is in critical section.\n", thread\_num);

leave\_critical\_section(thread\_num);

printf("Thread %d is out of critical section.\n", thread\_num);

}

pthread\_exit(NULL);

}

int main()

{

pthread\_t threads[NUM\_THREADS];

int thread\_args[NUM\_THREADS];

for (int i = 0; i < NUM\_THREADS; i++)

{

thread\_args[i] = i;

pthread\_create(&threads[i], NULL, thread\_function, &thread\_args[i]);

}

for (int i = 0; i < NUM\_THREADS; i++)

{

pthread\_join(threads[i], NULL);

}

return 0;

}

