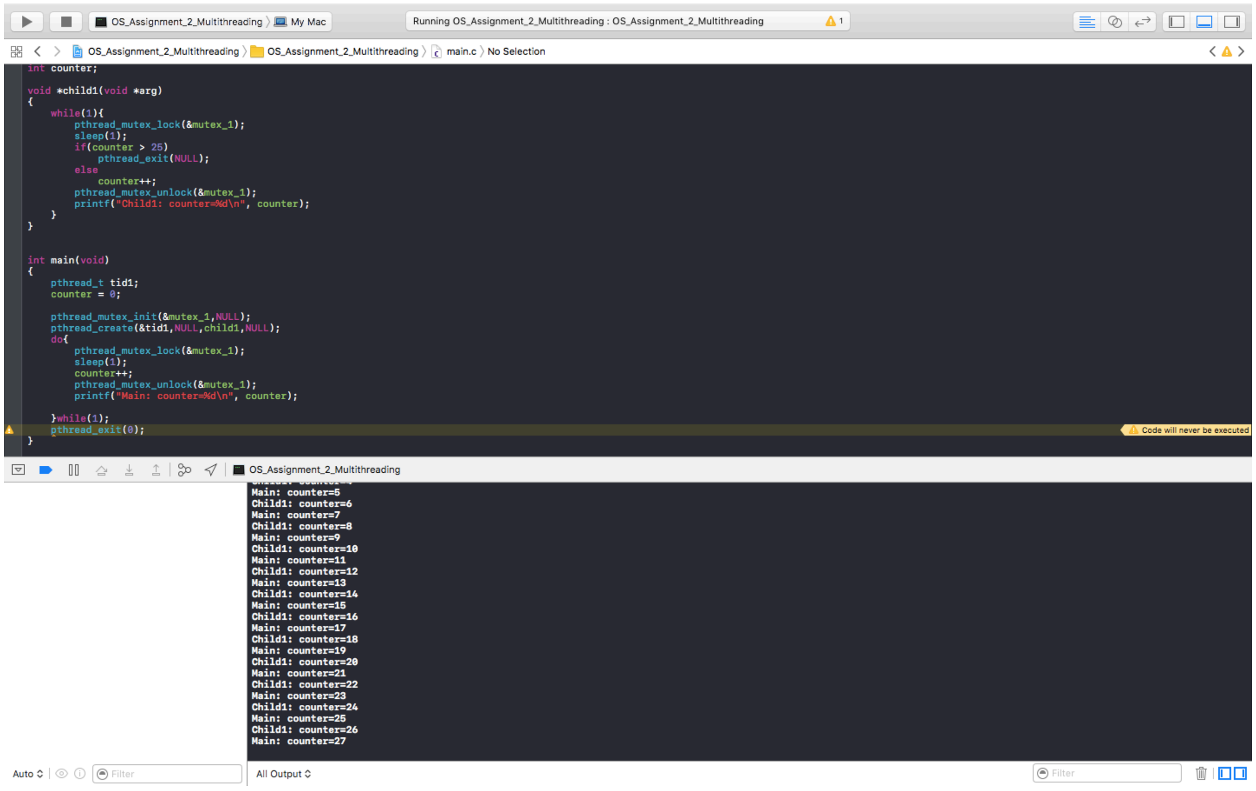
Question:

This program creates a new thread. Both the main thread and the new thread then increment the variable **counter** infinitely. After incrementing the value of counter, each thread prints a message showing the value of the variable. One of the threads exits when it finds that the counter value has exceeded 25. Run the program and explain the output. Why do the print statements stop appearing after a certain point in the program?

Answer:

The program first creates a child thread. The child thread and the main thread then iteratively increment a global counter. If the value of the counter exceeds 25, the child thread exits. However, when the value of the counter is equal to 25, the child thread increases the value of the counter and prints it (value being 26). The control then goes to the main thread where it gets incremented and printed (value being 27). When the control goes back to the child thread, as the counter value is greater than 25, the control exits. The program attached to the question gets stuck at this point. This is because in the program, the child process acquires a mutex when the thread starts execution, however it releases the mutex after increasing the counter value. When the value of the counter is greater than 25, the code that releases the mutex is not executed. This ensures that the control doesn't go back to the main program. As a result, the program gets stuck.


OS_Assignment_2_Multithreading 〉 My Mac
Running OS_Assignment_2_Multithreading : OS_Assignment_2_Multithreading
OS_Assignment_2_Multithreading 〉 OS_Assignment_2_Multithreading 〉 main.c 〉 No Selection
Code will never be executed
OS_Assignment_2_Multithreading
Auto
Filter
All Output
Filter

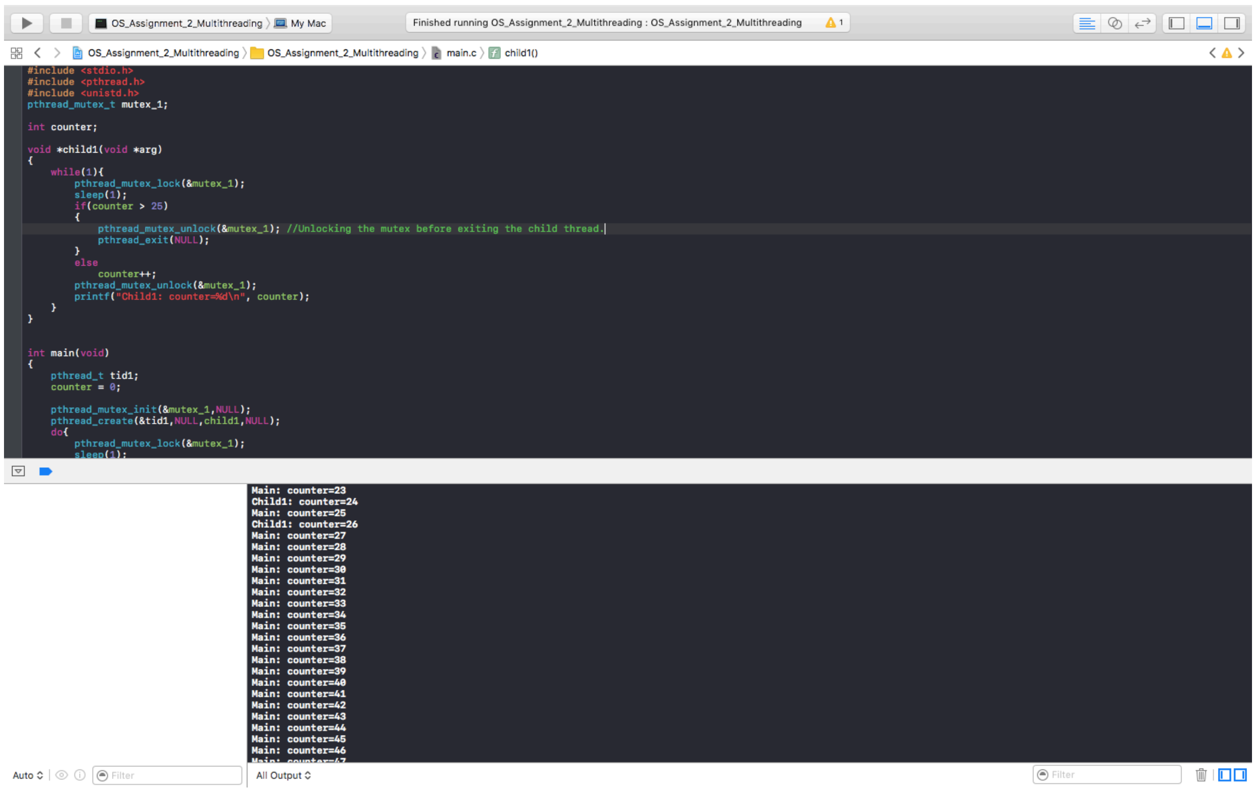
```c
int counter;

void *child1(void *arg)
{
    while(1){
        pthread_mutex_lock(&mutex_1);
        sleep(1);
        if(counter > 25)
            pthread_exit(NULL);
        else
            counter++;
        pthread_mutex_unlock(&mutex_1);
        printf("Child1: counter=%d\n", counter);
    }
}


int main(void)
{
    pthread_t tid1;
    counter = 0;

    pthread_mutex_init(&mutex_1,NULL);
    pthread_create(&tid1,NULL,child1,NULL);
    do{
        pthread_mutex_lock(&mutex_1);
        sleep(1);
        counter++;
        pthread_mutex_unlock(&mutex_1);
        printf("Main: counter=%d\n", counter);

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

```
Main: counter=5
Child1: counter=6
Main: counter=7
Child1: counter=8
Main: counter=9
Child1: counter=10
Main: counter=11
Child1: counter=12
Main: counter=13
Child1: counter=14
Main: counter=15
Child1: counter=16
Main: counter=17
Child1: counter=18
Main: counter=19
Child1: counter=20
Main: counter=21
Child1: counter=22
Main: counter=23
Child1: counter=24
Main: counter=25
Child1: counter=26
Main: counter=27
```

This problem can be fixed by releasing the mutex inside the if statement (Which checks if the counter value is greater than 25), before exiting. This is done by using the following statement.

```
pthread_mutex_unlock(&mutex_1);
```

The output after this change can be seen in the following snapshot. As can be seen, the counter in the child process stops incrementing after it prints 26. The counter of the main program however continues to be incremented and printed.


OS_Assignment_2_Multithreading
My Mac
Finished running OS_Assignment_2_Multithreading : OS_Assignment_2_Multithreading
OS_Assignment_2_Multithreading
OS_Assignment_2_Multithreading
main.c
child1()
Auto
Filter
All Output
Filter

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
pthread_mutex_t mutex_1;

int counter;

void *child1(void *arg)
{
    while(1){
        pthread_mutex_lock(&mutex_1);
        sleep(1);
        if(counter > 25)
        {
            pthread_mutex_unlock(&mutex_1); //Unlocking the mutex before exiting the child thread.
            pthread_exit(NULL);
        }
        else
            counter++;
        pthread_mutex_unlock(&mutex_1);
        printf("Child1: counter=%d\n", counter);
    }
}


int main(void)
{
    pthread_t tid1;
    counter = 0;

    pthread_mutex_init(&mutex_1,NULL);
    pthread_create(&tid1,NULL,child1,NULL);
    do{
        pthread_mutex_lock(&mutex_1);
        sleep(1);
```

```
Main: counter=23
Child1: counter=24
Main: counter=25
Child1: counter=26
Main: counter=27
Main: counter=28
Main: counter=29
Main: counter=30
Main: counter=31
Main: counter=32
Main: counter=33
Main: counter=34
Main: counter=35
Main: counter=36
Main: counter=37
Main: counter=38
Main: counter=39
Main: counter=40
Main: counter=41
Main: counter=42
Main: counter=43
Main: counter=44
Main: counter=45
Main: counter=46
Main: counter=47
```