



National Textile University

Department of Computer Science

Subject:

Operating system

Submitted To:

Sir Nasir

Submitted By:

Noor ul Ain

Registration No:

23-NTU-CS-1221

Home Task No:

10

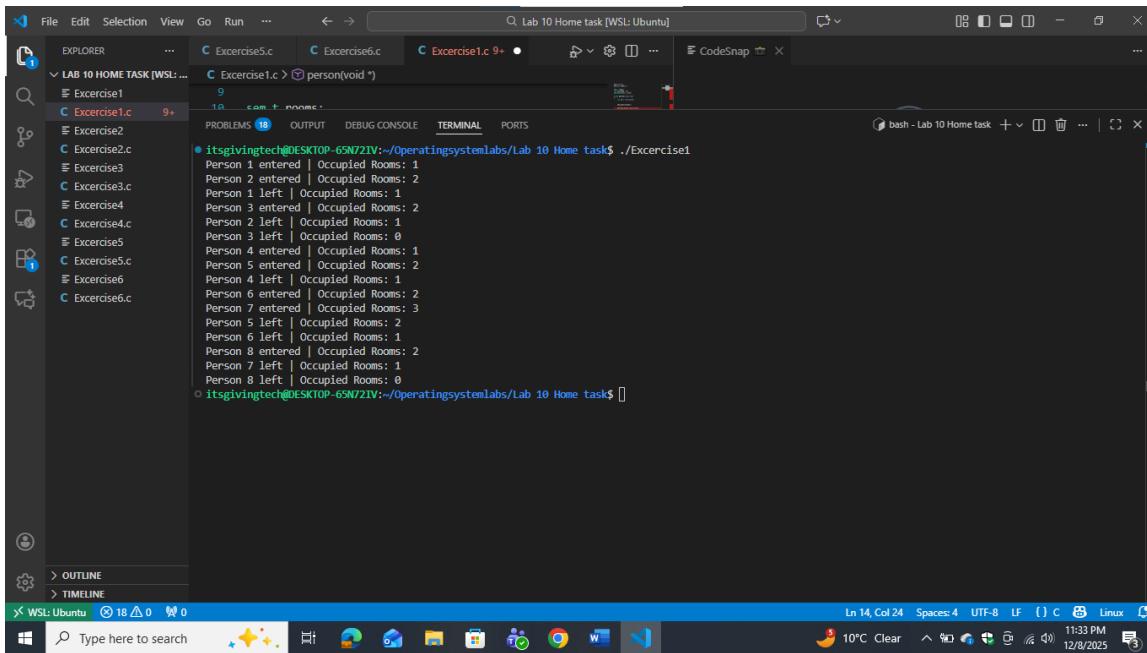
Semester:

5th

Exercise 1:

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define N 3
8 #define TOTAL 8
9
10 sem_t rooms;
11 int occupied = 0;
12 pthread_mutex_t lock;
13
14 void* person(void* num)
15 {
16     int id = *(int*)num;
17
18     sem_wait(&rooms);
19
20     pthread_mutex_lock(&lock);
21     occupied++;
22     printf("Person %d entered | Occupied Rooms: %d\n", id, occupied);
23     pthread_mutex_unlock(&lock);
24
25
26     sleep(rand() % 3 + 1);
27
28     pthread_mutex_lock(&lock);
29     occupied--;
30     printf("Person %d left | Occupied Rooms: %d\n", id, occupied);
31     pthread_mutex_unlock(&lock);
32
33
34     sem_post(&rooms);
35
36     return NULL;
37 }
38
39
40 int main()
41 {
42     pthread_t p[TOTAL];
43     int ids[TOTAL];
44
45     sem_init(&rooms, 0, N);
46     pthread_mutex_init(&lock, NULL);
47
48     for (int i = 0; i < TOTAL; i++)
49     {
50         ids[i] = i + 1;
51         pthread_create(&p[i], NULL, person, &ids[i]);
52         sleep(1);
53     }
54
55     for (int i = 0; i < TOTAL; i++)
56     {
57         pthread_join(p[i], NULL);
58     }
59
60     sem_destroy(&rooms);
61     pthread_mutex_destroy(&lock);
62
63     return 0;
64 }
```

Output:



The screenshot shows the Visual Studio Code interface running in a WSL Ubuntu terminal. The terminal window displays the output of a C program named Exercise1.c. The program simulates people entering and leaving rooms, printing the count of occupied rooms each time. The output is as follows:

```
Person 1 entered | Occupied Rooms: 1
Person 2 entered | Occupied Rooms: 2
Person 1 left | Occupied Rooms: 1
Person 3 entered | Occupied Rooms: 2
Person 2 left | Occupied Rooms: 1
Person 3 left | Occupied Rooms: 0
Person 4 entered | Occupied Rooms: 1
Person 5 entered | Occupied Rooms: 2
Person 4 left | Occupied Rooms: 1
Person 6 entered | Occupied Rooms: 2
Person 7 entered | Occupied Rooms: 3
Person 5 left | Occupied Rooms: 2
Person 6 left | Occupied Rooms: 1
Person 8 entered | Occupied Rooms: 2
Person 7 left | Occupied Rooms: 1
Person 8 left | Occupied Rooms: 0
```

Exercise 2:



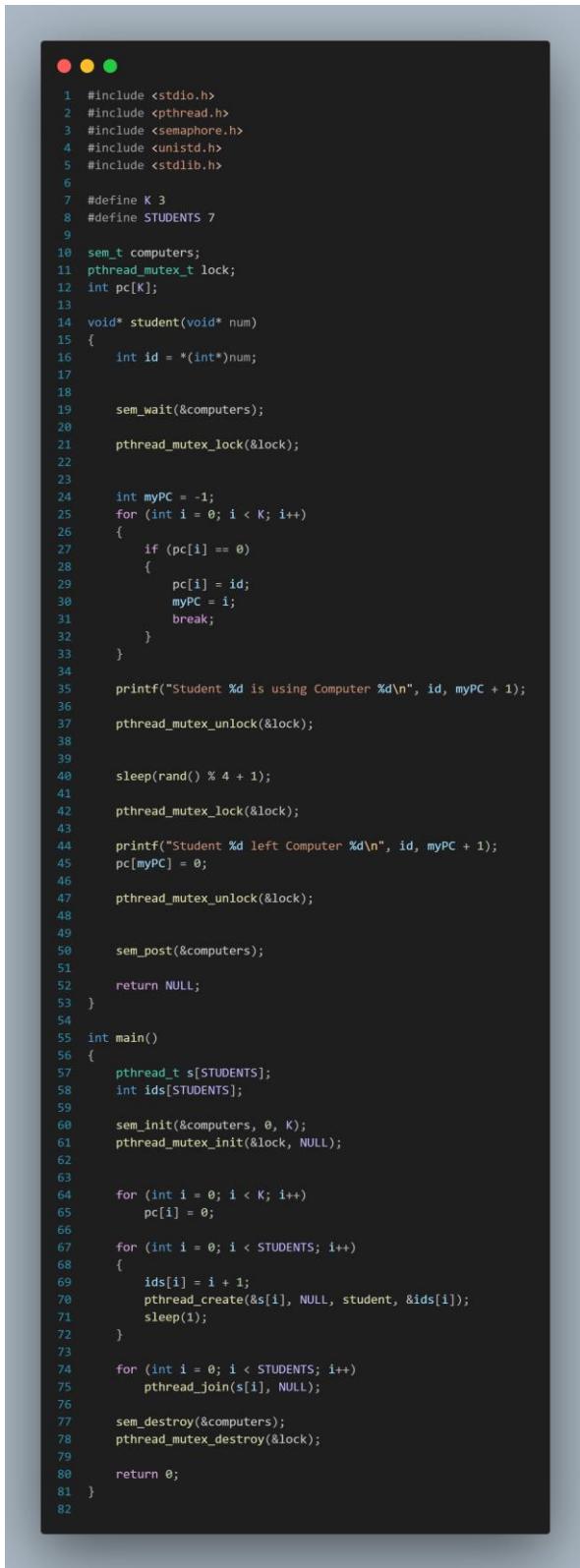
```
● ● ●
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define MAX_DOWNLOADS 3
8 #define TOTAL_FILES 8
9
10 sem_t downloadSlots;
11
12 void* download(void* num)
13 {
14     int id = *(int*)num;
15
16     sem_wait(&downloadSlots);
17     printf("Download %d started\n", id);
18
19     sleep(rand() % 5 + 1);
20     printf("Download %d finished\n", id);
21
22     sem_post(&downloadSlots);
23
24     return NULL;
25 }
26
27 int main()
28 {
29     pthread_t d[TOTAL_FILES];
30     int ids[TOTAL_FILES];
31
32     sem_init(&downloadSlots, 0, MAX_DOWNLOADS);
33
34     for (int i = 0; i < TOTAL_FILES; i++)
35     {
36         ids[i] = i + 1;
37         pthread_create(&d[i], NULL, download, &ids[i]);
38         sleep(1);
39     }
40
41     for (int i = 0; i < TOTAL_FILES; i++)
42     {
43         pthread_join(d[i], NULL);
44     }
45
46     sem_destroy(&downloadSlots);
47
48     return 0;
49 }
```

Output:

A screenshot of a terminal window in Visual Studio Code (VS Code) running in a Windows Subsystem for Linux (WSL) environment. The terminal is titled "Lab 10 Home task [WSL: Ubuntu]". The command run is "void* download(void* num)". The output shows a multi-threaded download process:

```
#1solvingtech@DESKTOP-65N7ZIV:~/Operatingsystems/labs/Lab 10 Home task$ ./Exercise2
12 void* download(void* num)
#1solvingtech@DESKTOP-65N7ZIV:~/Operatingsystems/labs/Lab 10 Home task$ ./Exercise2
Download 1 started
Download 2 started
Download 3 started
Download 4 started
Download 4 started
Download 1 finished
Download 4 finished
Download 5 started
Download 6 started
Download 8 started
Download 8 started
Download 6 finished
Download 7 finished
Download 8 finished
Download 8 finished
```

Exercise 3:



```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define K 3
8 #define STUDENTS 7
9
10 sem_t computers;
11 pthread_mutex_t lock;
12 int pc[K];
13
14 void* student(void* num)
15 {
16     int id = *(int*)num;
17
18     sem_wait(&computers);
19
20     pthread_mutex_lock(&lock);
21
22
23     int myPC = -1;
24     for (int i = 0; i < K; i++)
25     {
26         if (pc[i] == 0)
27         {
28             pc[i] = id;
29             myPC = i;
30             break;
31         }
32     }
33
34     printf("Student %d is using Computer %d\n", id, myPC + 1);
35
36     pthread_mutex_unlock(&lock);
37
38
39     sleep(rand() % 4 + 1);
40
41     pthread_mutex_lock(&lock);
42
43     printf("Student %d left Computer %d\n", id, myPC + 1);
44     pc[myPC] = 0;
45
46     pthread_mutex_unlock(&lock);
47
48
49     sem_post(&computers);
50
51     return NULL;
52 }
53
54 int main()
55 {
56     pthread_t s[STUDENTS];
57     int ids[STUDENTS];
58
59     sem_init(&computers, 0, K);
60     pthread_mutex_init(&lock, NULL);
61
62
63     for (int i = 0; i < K; i++)
64         pc[i] = 0;
65
66     for (int i = 0; i < STUDENTS; i++)
67     {
68         ids[i] = i + 1;
69         pthread_create(&s[i], NULL, student, &ids[i]);
70         sleep(1);
71     }
72
73     for (int i = 0; i < STUDENTS; i++)
74         pthread_join(s[i], NULL);
75
76     sem_destroy(&computers);
77     pthread_mutex_destroy(&lock);
78
79     return 0;
80 }
81
82 }
```

Output:

The screenshot shows the Visual Studio Code interface running on a Windows host with a Linux workspace (WSL: Ubuntu). The code editor displays a file named `Excercise3.c` containing C code for a producer-consumer problem involving semaphores and threads. The terminal window shows the execution of the program, which prints messages indicating student activity and computer usage. The status bar at the bottom provides system information like temperature and date.

```
#include <stdio.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdlib.h>

#define K 3
#define STUDENTS 7

sem_t computers;
pthread_mutex_t lock;
int pc[K];

void* student(void* num)
```

```
itsgivingtech@DESKTOP-6SN72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Excercise3
Student 1 is using Computer 1
Student 2 is using Computer 2
Student 3 is using Computer 3
Student 1 left Computer 1
Student 4 is using Computer 1
Student 2 left Computer 2
Student 3 left Computer 3
Student 5 is using Computer 2
Student 6 is using Computer 3
Student 5 left Computer 2
Student 7 is using Computer 2
Student 4 left Computer 1
Student 6 left Computer 3
Student 7 left Computer 2
```

11:40 PM 10°C Clear 12/8/2025

Exercise 4:

```
 1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define WORKERS 3
8 #define TASKS 10
9
10 sem_t workerPool;
11
12 void* task(void* num)
13 {
14     int id = *(int*)num;
15
16     sem_wait(&workerPool);
17     printf("Task %d is being processed by a worker\n", id);
18
19     sleep(rand() % 2 + 1);
20
21     printf("Task %d is finished\n", id);
22
23     sem_post(&workerPool);
24
25     return NULL;
26 }
27
28 int main()
29 {
30     pthread_t t[TASKS];
31     int ids[TASKS];
32
33     sem_init(&workerPool, 0, WORKERS);
34
35     for (int i = 0; i < TASKS; i++)
36     {
37         ids[i] = i + 1;
38         pthread_create(&t[i], NULL, task, &ids[i]);
39         sleep(1);
40     }
41
42     for (int i = 0; i < TASKS; i++)
43     {
44         pthread_join(t[i], NULL);
45     }
46
47     sem_destroy(&workerPool);
48
49     return 0;
50 }
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home tasks$ gcc Exercise4.c -o Exercise4
● itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home tasks$ ./Exercise4
Task 1 is being processed by a worker
Task 1 is finished
Task 2 is being processed by a worker
Task 2 is finished
Task 3 is being processed by a worker
Task 3 is finished
Task 4 is being processed by a worker
Task 4 is finished
Task 5 is being processed by a worker
Task 5 is finished
Task 6 is being processed by a worker
Task 6 is finished
Task 7 is being processed by a worker
Task 7 is finished
Task 8 is being processed by a worker
Task 8 is finished
Task 9 is being processed by a worker
Task 9 is finished
Task 10 is being processed by a worker
Task 10 is finished
● itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home tasks$
```

Exercise 5:

```
● ● ●

1 #include <stdio.h>
2 #include <pthread.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5
6 #define STATIONS 2
7 #define CARS 8
8
9 sem_t washStations;
10 pthread_mutex_t lock;
11 int waiting = 0;
12
13 void* car(void* num)
14 {
15     int id = *(int*)num;
16
17     pthread_mutex_lock(&lock);
18     waiting++;
19     printf("Car %d is waiting | Queue: %d\n", id, waiting);
20     pthread_mutex_unlock(&lock);
21
22     sem_wait(&washStations);
23
24     pthread_mutex_lock(&lock);
25     waiting--;
26     printf("Car %d is being washed | Queue: %d\n", id, waiting);
27     pthread_mutex_unlock(&lock);
28
29
30     sleep(3);
31
32     printf("Car %d has finished washing\n", id);
33
34
35     sem_post(&washStations);
36
37     return NULL;
38 }
39
40 int main()
41 {
42     pthread_t c[CARS];
43     int ids[CARS];
44
45     sem_init(&washStations, 0, STATIONS);
46     pthread_mutex_init(&lock, NULL);
47
48     for (int i = 0; i < CARS; i++)
49     {
50         ids[i] = i + 1;
51         pthread_create(&c[i], NULL, car, &ids[i]);
52         sleep(1);
53     }
54
55     for (int i = 0; i < CARS; i++)
56         pthread_join(c[i], NULL);
57
58     sem_destroy(&washStations);
59     pthread_mutex_destroy(&lock);
60
61     return 0;
62 }
63 }
```

Output:

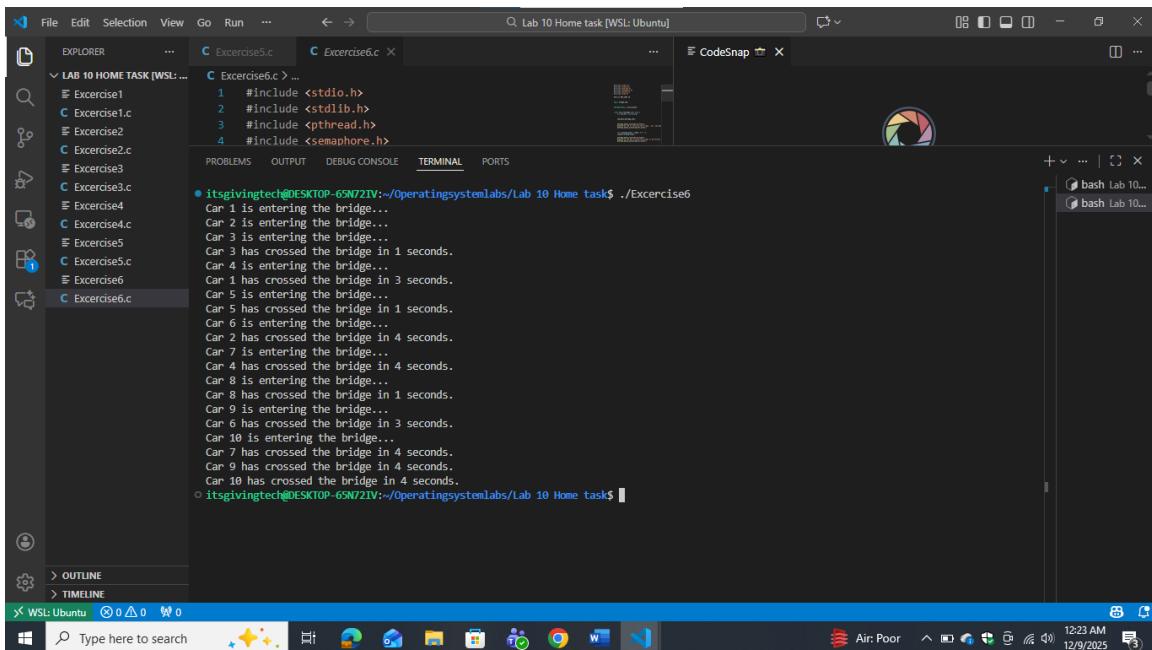
```
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$ ./Exercise5
Car 1 is waiting | Queue: 1
Car 1 is being washed | Queue: 0
Car 2 is waiting | Queue: 1
Car 2 is being washed | Queue: 0
Car 3 is waiting | Queue: 1
Car 3 has finished washing
Car 3 is being washed | Queue: 0
Car 4 is waiting | Queue: 1
Car 4 has finished washing
Car 4 is being washed | Queue: 0
Car 5 is waiting | Queue: 1
Car 5 has finished washing
Car 5 is being washed | Queue: 2
Car 3 has finished washing
Car 5 is being washed | Queue: 1
Car 7 is waiting | Queue: 2
Car 4 has finished washing
Car 6 is being washed | Queue: 1
Car 8 is waiting | Queue: 2
Car 5 has finished washing
Car 7 is being washed | Queue: 1
Car 6 has finished washing
Car 8 is being washed | Queue: 0
Car 7 has finished washing
Car 8 has finished washing
itsgivingtech@DESKTOP-65N72IV:~/Operatingsystemlabs/Lab 10 Home task$
```

Exercise 6:



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <semaphore.h>
5 #include <unistd.h>
6 #include <time.h>
7
8 #define MAX_CARS 10
9
10 sem_t bridge_sem;
11
12
13 pthread_mutex_t print_mutex;
14
15
16
17 void* car_crossing(void* arg) {
18     int car_id = *(int*)arg;
19
20     sem_wait(&bridge_sem);
21
22
23     pthread_mutex_lock(&print_mutex);
24     printf("Car %d is entering the bridge...\n", car_id);
25     pthread_mutex_unlock(&print_mutex);
26
27
28     int crossing_time = rand() % 4 + 1;
29     sleep(crossing_time);
30
31     pthread_mutex_lock(&print_mutex);
32     printf("Car %d has crossed the bridge in %d seconds.\n", car_id, crossing_time);
33     pthread_mutex_unlock(&print_mutex);
34
35
36     sem_post(&bridge_sem);
37
38     pthread_exit(NULL);
39 }
40
41 int main() {
42     srand(time(NULL));
43
44     pthread_t cars[MAX_CARS];
45     int car_ids[MAX_CARS];
46
47
48     sem_init(&bridge_sem, 0, 3);
49
50
51     pthread_mutex_init(&print_mutex, NULL);
52
53
54     for (int i = 0; i < MAX_CARS; i++) {
55         car_ids[i] = i + 1;
56         pthread_create(&cars[i], NULL, car_crossing, &car_ids[i]);
57         usleep(200000);
58     }
59
60
61     for (int i = 0; i < MAX_CARS; i++) {
62         pthread_join(cars[i], NULL);
63     }
64
65
66     sem_destroy(&bridge_sem);
67     pthread_mutex_destroy(&print_mutex);
68
69
70     return 0;
71 }
72 }
```

Output:



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

int sem_id[10];
int car_id[10];
int count = 0;
int max_cars = 10;
int current_car = 0;
int time = 0;

void* cross_bridge(void* arg) {
    int id = *(int*)arg;
    while (current_car < max_cars) {
        if (sem_getvalue(sem_id[current_car], &count) == 1) {
            printf("Car %d is entering the bridge...\n", id);
            sleep(1);
            if (current_car == 0) {
                printf("Car %d has crossed the bridge in %d seconds.\n", id, time);
                time++;
            } else {
                printf("Car %d has crossed the bridge in %d seconds.\n", id, time - 1);
            }
            current_car++;
        }
    }
}

int main() {
    for (int i = 0; i < max_cars; i++) {
        sem_id[i] = sem_init(1, 1, 1);
        car_id[i] = i;
    }
    for (int i = 0; i < max_cars; i++) {
        pthread_t thread;
        pthread_create(&thread, NULL, cross_bridge, &car_id[i]);
    }
    for (int i = 0; i < max_cars; i++) {
        pthread_join(thread, NULL);
    }
    for (int i = 0; i < max_cars; i++) {
        sem_destroy(sem_id[i]);
    }
}
```