



# 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

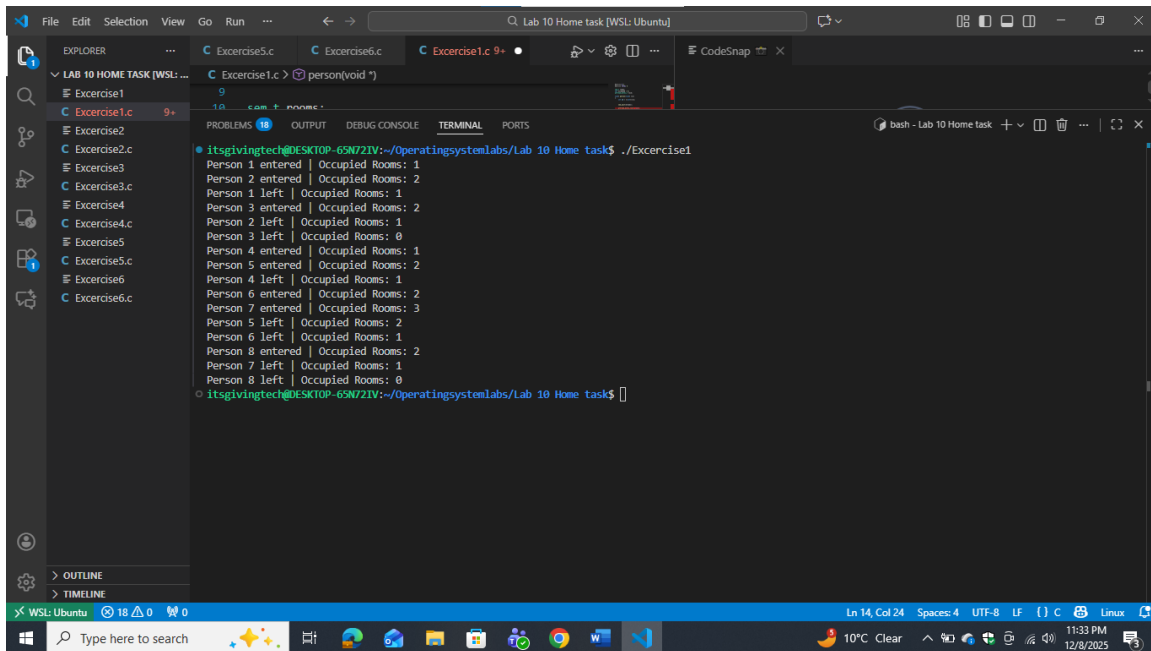
**Semester:**

5<sup>th</sup>

## 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     sleep(rand() % 3 + 1);
26
27     pthread_mutex_lock(&lock);
28     occupied--;
29     printf("Person %d left | Occupied Rooms: %d\n", id, occupied);
30     pthread_mutex_unlock(&lock);
31
32     sem_post(&rooms);
33
34     return NULL;
35 }
36
37 int main()
38 {
39     pthread_t p[TOTAL];
40     int ids[TOTAL];
41
42     sem_init(&rooms, 0, N);
43     pthread_mutex_init(&lock, NULL);
44
45     for (int i = 0; i < TOTAL; i++)
46     {
47         ids[i] = i + 1;
48         pthread_create(&p[i], NULL, person, &ids[i]);
49         sleep(1);
50     }
51
52     for (int i = 0; i < TOTAL; i++)
53     {
54         pthread_join(p[i], NULL);
55     }
56
57     sem_destroy(&rooms);
58     pthread_mutex_destroy(&lock);
59
60     return 0;
61 }
62
63
64
65
```

Output:



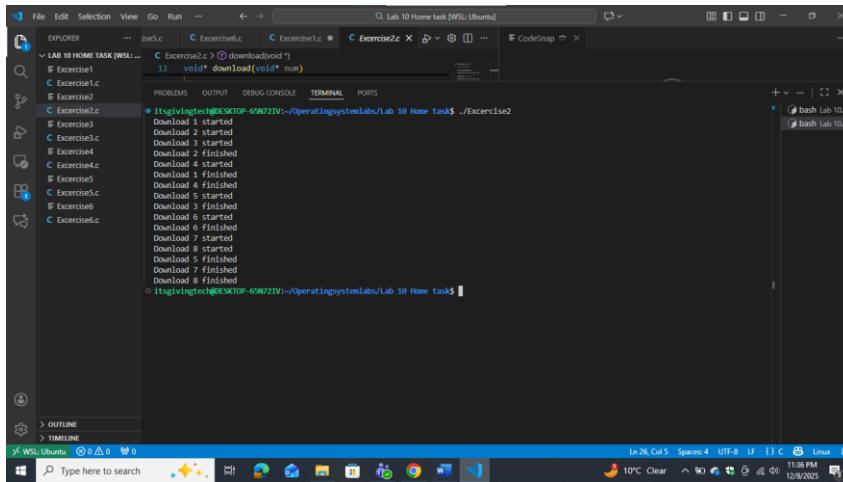
The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project named 'LAB 10 HOME TASK [WSL: Ubuntu]' with several C files. The terminal window displays the output of a C program named 'Exercise1.c'. The program's output is a log of person entries and room occupancy counts.

```
itsgivingtech@DESKTOP-65N72IV:~/operatingsystemlabs/Lab 10 Home task$ ./Exercise1
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
itsgivingtech@DESKTOP-65N72IV:~/operatingsystemlabs/Lab 10 Home task$
```

## 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
21     printf("Download %d finished\n", id);
22
23     sem_post(&downloadSlots);
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 }
50
51
```

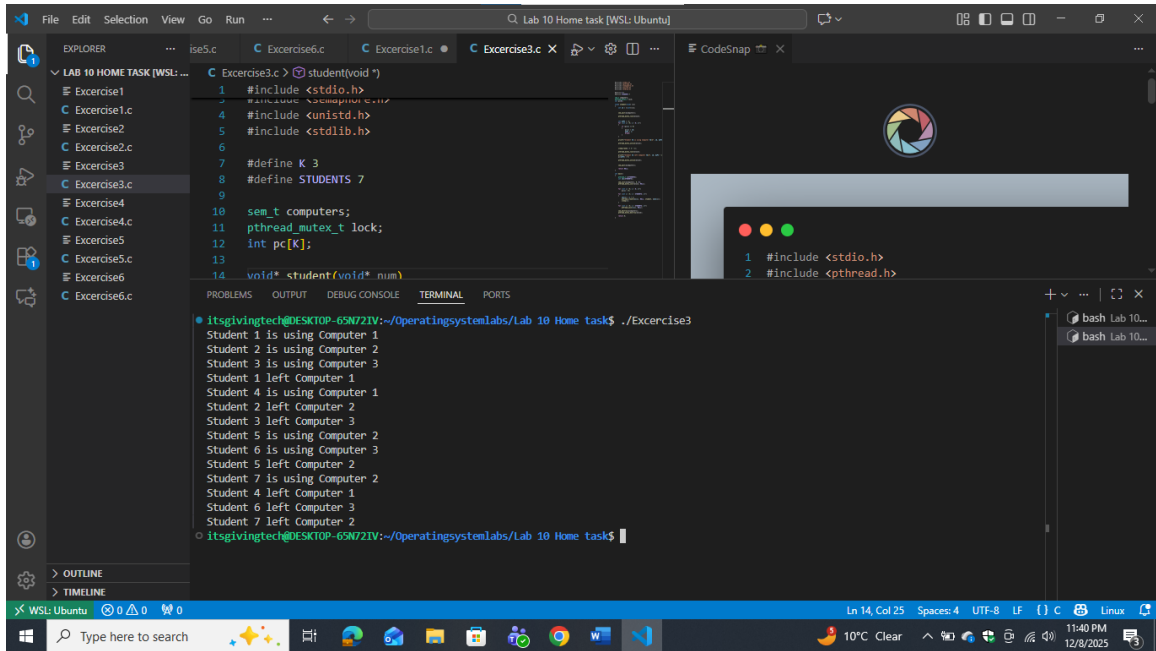
Output:



### 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     int myPC = -1;
23     for (int i = 0; i < K; i++)
24     {
25         if (pc[i] == 0)
26         {
27             pc[i] = id;
28             myPC = i;
29             break;
30         }
31     }
32
33     printf("Student %d is using Computer %d\n", id, myPC + 1);
34
35     pthread_mutex_unlock(&lock);
36
37     sleep(rand() % 4 + 1);
38
39     pthread_mutex_lock(&lock);
40
41     printf("Student %d left Computer %d\n", id, myPC + 1);
42     pc[myPC] = 0;
43
44     pthread_mutex_unlock(&lock);
45
46     sem_post(&computers);
47
48     return NULL;
49 }
50
51 int main()
52 {
53     pthread_t s[STUDENTS];
54     int ids[STUDENTS];
55
56     sem_init(&computers, 0, K);
57     pthread_mutex_init(&lock, NULL);
58
59     for (int i = 0; i < K; i++)
60         pc[i] = 0;
61
62     for (int i = 0; i < STUDENTS; i++)
63     {
64         ids[i] = i + 1;
65         pthread_create(&s[i], NULL, student, &ids[i]);
66         sleep(1);
67     }
68
69     for (int i = 0; i < STUDENTS; i++)
70         pthread_join(s[i], NULL);
71
72     sem_destroy(&computers);
73     pthread_mutex_destroy(&lock);
74
75     return 0;
76 }
77
78
79
80
81
82
```

Output:



```
1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <unistd.h>
4 #include <stdlib.h>
5
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)
```

```
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 10 Home task$ ./Exercise3
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
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 10 Home task$
```

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
17     sem_wait(&workerPool);
18
19     printf("Task %d is being processed by a worker\n", id);
20
21
22     sleep(rand() % 2 + 1);
23
24     printf("Task %d is finished\n", id);
25
26
27     sem_post(&workerPool);
28
29     return NULL;
30 }
31
32 int main()
33 {
34     pthread_t t[TASKS];
35     int ids[TASKS];
36
37
38     sem_init(&workerPool, 0, WORKERS);
39
40     for (int i = 0; i < TASKS; i++)
41     {
42         ids[i] = i + 1;
43         pthread_create(&t[i], NULL, task, &ids[i]);
44         sleep(1);
45     }
46
47     for (int i = 0; i < TASKS; i++)
48     {
49         pthread_join(t[i], NULL);
50     }
51
52     sem_destroy(&workerPool);
53
54     return 0;
55 }
56

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• itsgivingtech@DESKTOP-69W72IV:~/OperatingSystemLabs/Lab 10 Home task$ gcc Exercised4.c -o Exercised4
• itsgivingtech@DESKTOP-69W72IV:~/OperatingSystemLabs/Lab 10 Home task$ ./Exercised4
Task 1 is being processed by a worker
Task 2 is being processed by a worker
Task 1 is finished
Task 2 is finished
Task 3 is being processed by a worker
Task 4 is being processed by a worker
Task 3 is finished
Task 5 is being processed by a worker
Task 4 is finished
Task 6 is being processed by a worker
Task 5 is finished
Task 7 is being processed by a worker
Task 6 is finished
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 10 is being processed by a worker
Task 9 is finished
Task 10 is finished
itsgivingtech@DESKTOP-69W72IV:~/OperatingSystemLabs/Lab 10 Home task$

```



## 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
23     sem_wait(&washStations);
24
25     pthread_mutex_lock(&lock);
26     waiting--;
27     printf("Car %d is being washed | Queue: %d\n", id, waiting);
28     pthread_mutex_unlock(&lock);
29
30
31     sleep(3);
32
33     printf("Car %d has finished washing\n", id);
34
35
36     sem_post(&washStations);
37
38     return NULL;
39 }
40
41 int main()
42 {
43     pthread_t c[CARS];
44     int ids[CARS];
45
46     sem_init(&washStations, 0, STATIONS);
47     pthread_mutex_init(&lock, NULL);
48
49     for (int i = 0; i < CARS; i++)
50     {
51         ids[i] = i + 1;
52         pthread_create(&c[i], NULL, car, &ids[i]);
53         sleep(1);
54     }
55
56     for (int i = 0; i < CARS; i++)
57         pthread_join(c[i], NULL);
58
59     sem_destroy(&washStations);
60     pthread_mutex_destroy(&lock);
61
62     return 0;
63 }
64
```

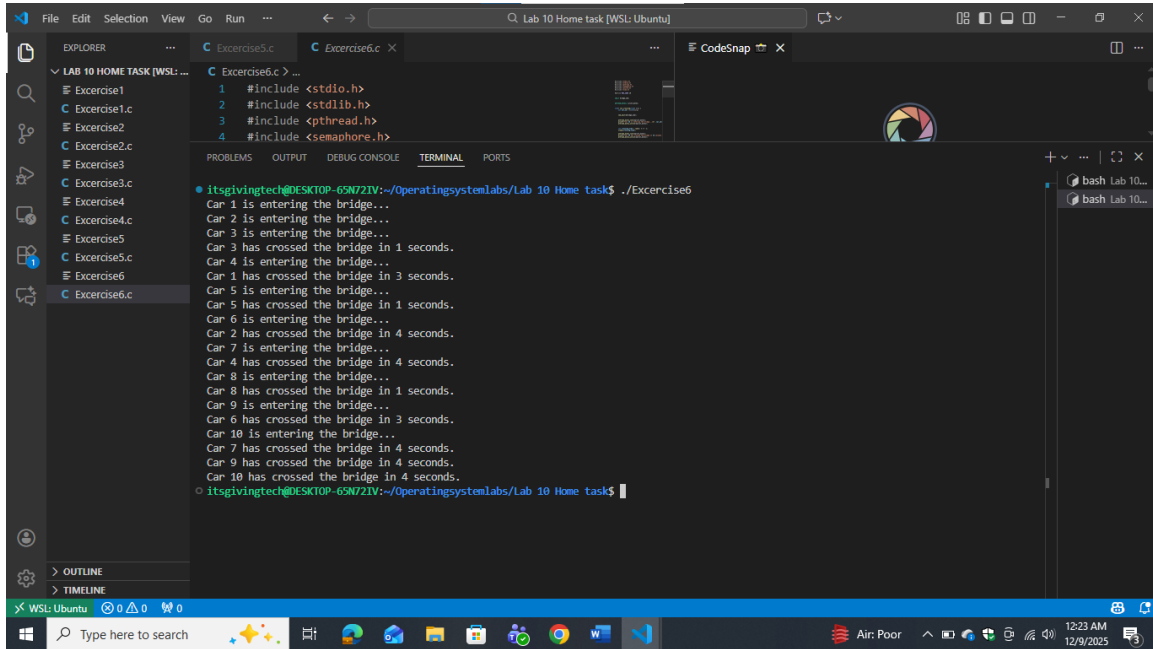
Output:

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

Output:



The screenshot shows a Visual Studio Code editor window titled "Lab 10 Home task [WSL: Ubuntu]". The Explorer sidebar on the left lists files under "LAB 10 HOME TASK [WSL: ...]": Exercise1, Exercise1.c, Exercise2, Exercise2.c, Exercise3, Exercise3.c, Exercise4, Exercise4.c, Exercise5, Exercise5.c, Exercise6, and Exercise6.c. The file "Exercise6.c" is selected and open in the editor. The code in the editor is:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <semaphore.h>
```

The TERMINAL tab at the bottom shows the output of running the program:

```
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 10 Home task$ ./Exercise6
Car 1 is entering the bridge...
Car 2 is entering the bridge...
Car 3 is entering the bridge...
Car 3 has crossed the bridge in 1 seconds.
Car 4 is entering the bridge...
Car 1 has crossed the bridge in 3 seconds.
Car 5 is entering the bridge...
Car 5 has crossed the bridge in 1 seconds.
Car 6 is entering the bridge...
Car 2 has crossed the bridge in 4 seconds.
Car 7 is entering the bridge...
Car 4 has crossed the bridge in 4 seconds.
Car 8 is entering the bridge...
Car 8 has crossed the bridge in 1 seconds.
Car 9 is entering the bridge...
Car 6 has crossed the bridge in 3 seconds.
Car 10 is entering the bridge...
Car 7 has crossed the bridge in 4 seconds.
Car 9 has crossed the bridge in 4 seconds.
Car 10 has crossed the bridge in 4 seconds.
itsgivingtech@DESKTOP-65N72IV:~/OperatingSystemLabs/Lab 10 Home task$
```

The Windows taskbar at the bottom shows the system clock as 12:23 AM on 12/9/2025.