

Operating Systems

LAB#11



23K-2001

BCS-4J

Q1:

Ubuntu 24.04.1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

May 4 21:36

Q1_k232001.c
~/Desktop/OS Lab Tasks/Lab10_k232001

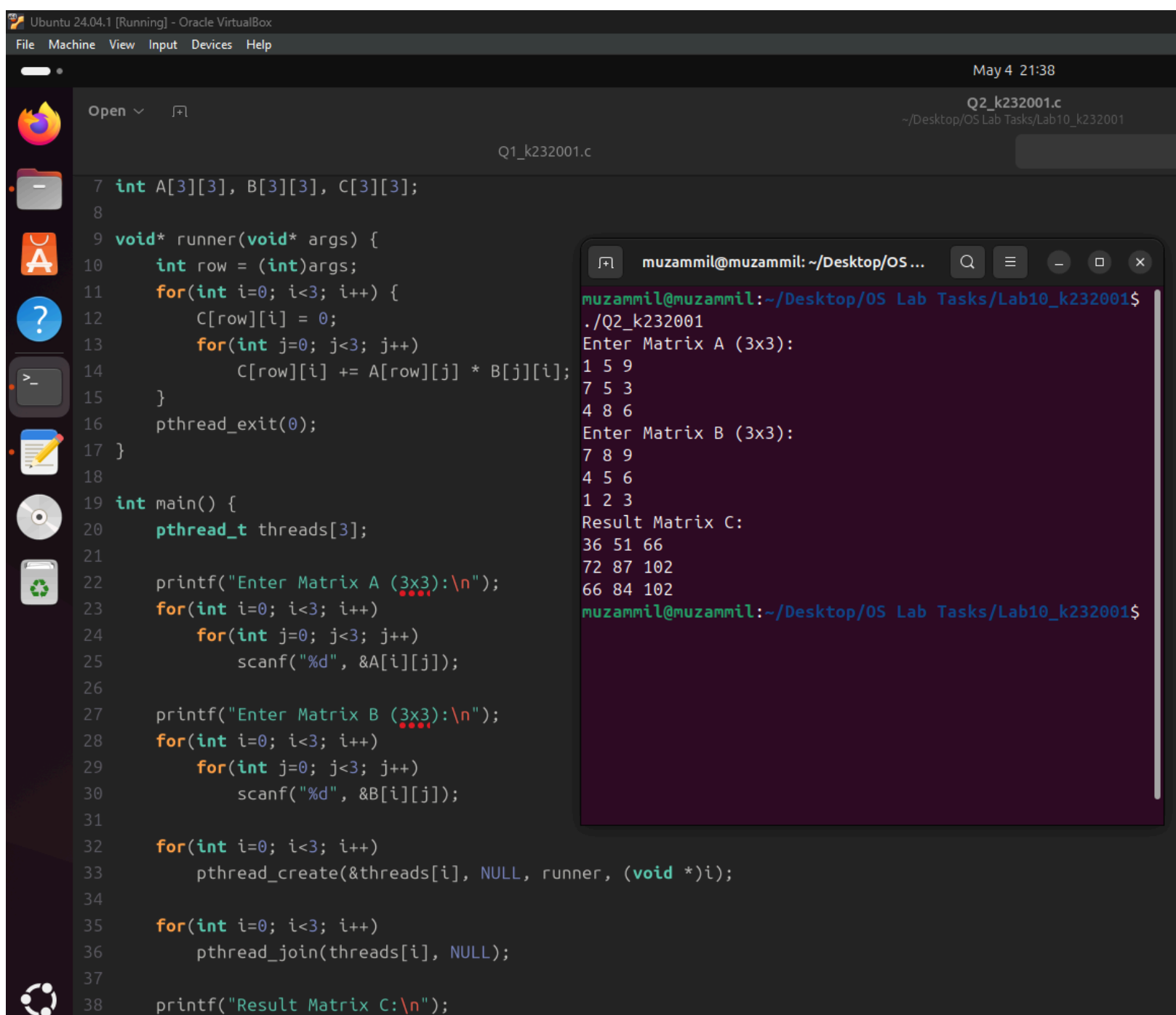
Open

```
27 void* runner(void* args){
28     int part = *(int*)args;
29     int mid = n/2;
30     if(part == 0)
31         sort(array, 0, mid-1);
32     else
33         sort(array, mid, n-1);
34     pthread_exit(0);
35 }
36
37 void merge(){
38     int *temp = malloc(sizeof(int)*n);
39     int i=0, j=n/2, k=0;
40     while(i<n/2 && j<n){
41         if(array[i] < array[j])
42             temp[k++] = array[i++];
43         else
44             temp[k++] = array[j++];
45     }
46
47     while(i<n/2)
48         temp[k++] = array[i++];
49
50     while(j<n)
51         temp[k++] = array[j++];
52
53     for(i=0; i<n; i++)
54         array[i] = temp[i];
55     free(temp);
56 }
57
58 int main() {
59     pthread_t t1, t2;
60     printf("Enter size for array: ");
61     scanf("%d", &n);
62     array = malloc(sizeof(int)*n);
63     init_array(array, n);
64
65     int part1 = 0, part2 = 1;
66     pthread_create(&t1, NULL, runner, &part1);
```

muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001

```
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$ ./Q1_k232001
Enter size for array: 55
Sorted array: 2 11 11 13 15 15 19 21 21 22 23 24 26 26 26 27 29 29 29 30 35 35 3
6 37 40 42 49 56 56 58 59 62 62 63 67 67 67 68 69 70 72 73 77 80 82 83 84 86 86
90 91 92 93 93 98
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Q2:



The image shows a screenshot of a virtual machine environment (Oracle VM VirtualBox) running Ubuntu 24.04.1. The main window is a code editor displaying a C program named `Q1_k232001.c`. The program is a multi-threaded matrix multiplication application. It defines a `runner` function that takes a row index as an argument and calculates the product of two 3x3 matrices, A and B, for a specific row. The `main` function initializes three threads, each running the `runner` function on a different row of matrix A. It then prints the resulting matrix C.

```
7 int A[3][3], B[3][3], C[3][3];
8
9 void* runner(void* args) {
10     int row = (int)args;
11     for(int i=0; i<3; i++) {
12         C[row][i] = 0;
13         for(int j=0; j<3; j++)
14             C[row][i] += A[row][j] * B[j][i];
15     }
16     pthread_exit(0);
17 }
18
19 int main() {
20     pthread_t threads[3];
21
22     printf("Enter Matrix A (3x3):\n");
23     for(int i=0; i<3; i++)
24         for(int j=0; j<3; j++)
25             scanf("%d", &A[i][j]);
26
27     printf("Enter Matrix B (3x3):\n");
28     for(int i=0; i<3; i++)
29         for(int j=0; j<3; j++)
30             scanf("%d", &B[i][j]);
31
32     for(int i=0; i<3; i++)
33         pthread_create(&threads[i], NULL, runner, (void *)i);
34
35     for(int i=0; i<3; i++)
36         pthread_join(threads[i], NULL);
37
38     printf("Result Matrix C:\n");
```

An inset terminal window shows the execution of the program. The user runs `./Q2_k232001` in the directory `~/Desktop/OS Lab Tasks/Lab10_k232001`. The terminal prompts for Matrix A and Matrix B, and displays the resulting Matrix C.

```
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$ ./Q2_k232001
Enter Matrix A (3x3):
1 5 9
7 5 3
4 8 6
Enter Matrix B (3x3):
7 8 9
4 5 6
1 2 3
Result Matrix C:
36 51 66
72 87 102
66 84 102
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Q3:

File Machine View Input Devices Help

May 4 21:53

Q3_k232001.c

```
7 //src: /home/muzammil/Desktop/OS Lab Tasks/Lab10_k232001/Q3TestSrc.txt
8 //dest: /home/muzammil/Desktop/OS Lab Tasks/Lab10_k232001/Q3TestDest.txt
9 int main(){
10     char src_path[256], dest_path[256];
11     printf("Enter source file path: ");
12     gets(src_path);
13     printf("Enter destination file path: ");
14     gets(dest_path);
15
16     int src = open(src_path, O_RDONLY);
17     if(src < 0){
18         perror("Error opening file src file: ");
19         printf("%s",src_path);
20         exit(1);
21     }
22
23     int dest = open(dest_path, O_WRONLY | O_CREAT, 0644);
24     if(dest < 0){
25         perror("Error opening file dest file.");
26         close(src);
27         exit(1);
28     }
29
30     char buffer[1024];
31     ssize_t bytes;
32     while((bytes = read(src, buffer, sizeof(buffer))) > 0){
33         if(write(dest, buffer, bytes) != bytes){
34             perror("Error writing to dest file.");
35             close(src);
36             close(dest);
37             exit(1);
38         }
39     }
```

muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001\$./Q3_k232001

Enter source file path: /home/muzammil/Desktop/OS Lab Tasks/Lab10_k232001/Q3TestSrc.txt

Enter destination file path: /home/muzammil/Desktop/OS Lab Tasks/Lab10_k232001/Q3TestDest.txt

Copying complete!

muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001\$ cat Q3TestSrc.txt

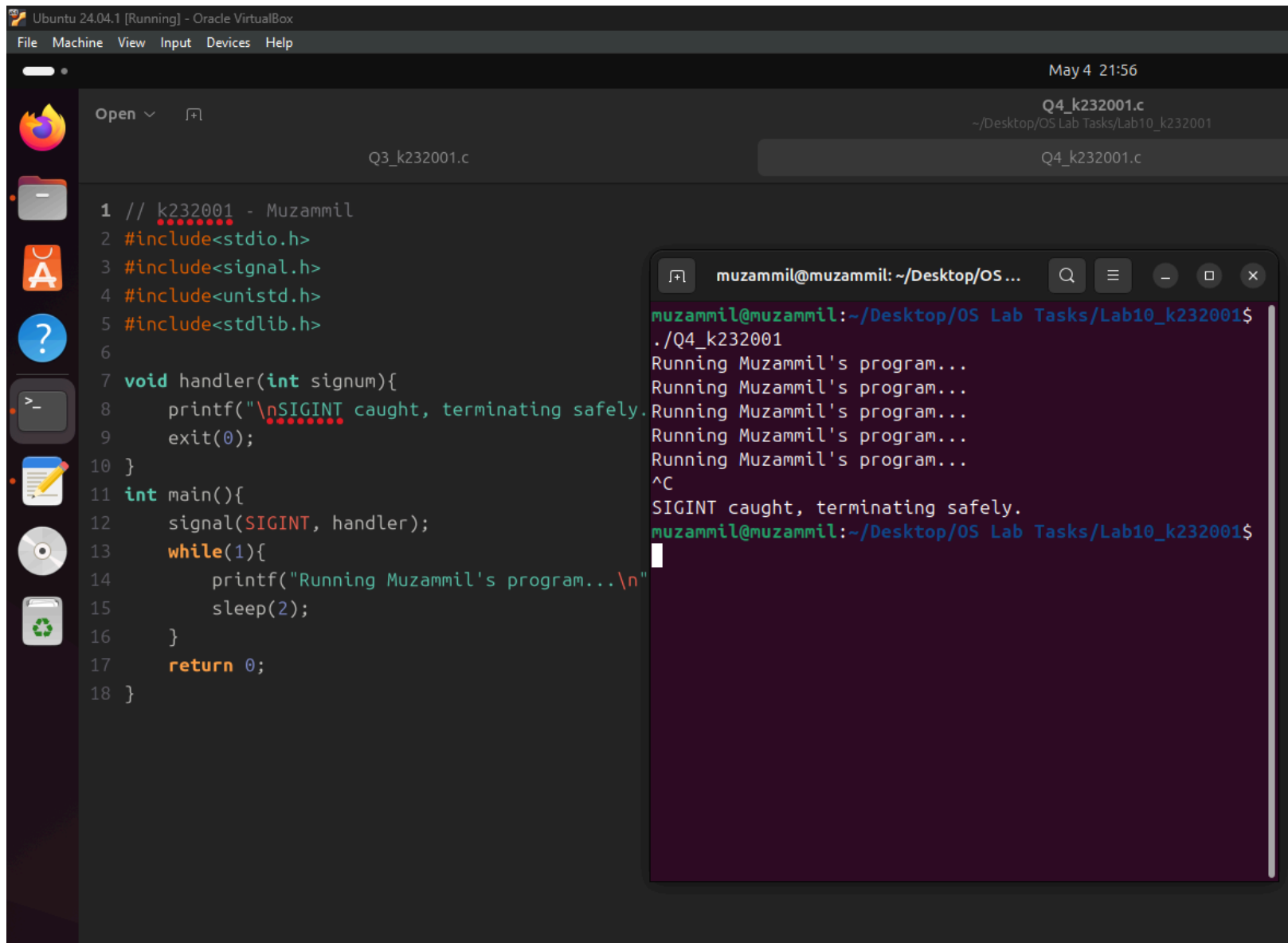
Hi this is a sample text file for question 3 to be read from.

muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001\$ cat Q3TestDest.txt

Hi this is a sample text file for question 3 to be read from.

muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001\$

Q4:



The screenshot shows a Linux desktop environment with a dark theme. The top bar indicates 'Ubuntu 24.04.1 [Running] - Oracle VirtualBox'. The main window is a code editor displaying a C program named 'Q3_k232001.c'. The code includes headers for `stdio.h`, `signal.h`, `unistd.h`, and `stdlib.h`. It defines a `handler` function for `SIGINT` that prints a message and calls `exit(0)`. The `main` function sets up the signal handler and enters a `while(1)` loop, printing 'Running Muzammil's program...' and sleeping for 2 seconds.

Overlaid on the bottom right is a terminal window titled 'muzammil@muzammil: ~/Desktop/OS ...'. The terminal shows the execution of the program: `./Q4_k232001` is run, resulting in five lines of 'Running Muzammil's program...'. Pressing `^C` (Ctrl+C) triggers the signal handler, resulting in the message 'SIGINT caught, terminating safely.' and the program's termination.

```
1 // k232001 - Muzammil
2 #include<stdio.h>
3 #include<signal.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6
7 void handler(int signum){
8     printf("\nSIGINT caught, terminating safely.
9     exit(0);
10 }
11 int main(){
12     signal(SIGINT, handler);
13     while(1){
14         printf("Running Muzammil's program...\n"
15         sleep(2);
16     }
17     return 0;
18 }
```

muzammil@muzammil: ~/Desktop/OS ...

muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001\$./Q4_k232001

Running Muzammil's program...

Running Muzammil's program...

Running Muzammil's program...

Running Muzammil's program...

Running Muzammil's program...

^C

SIGINT caught, terminating safely.

muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001\$

Q5:

Ubuntu 24.04.1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

May 4 21:58

Q5_k232001.c
~/Desktop/OS Lab Tasks/Lab10_k232001

Q4_k232001.c Q5_k232001.c

```
1 // k232001 - Muzammil
2 #include<stdio.h>
3 #include<signal.h>
4 #include<unistd.h>
5 #include<stdlib.h>
6
7 volatile sig_atomic_t paused = 0;
8
9 void handle_sigint(int signum){
10     paused = 1;
11     printf("\nTimer paused.\n");
12 }
13
14 void handle_sigtstp(int signum){
15     paused = 0;
16     printf("\nTimer resumed.\n");
17 }
18
19 int main(){
20     signal(SIGINT, handle_sigint);
21     signal(SIGTSTP, handle_sigtstp);
22     int i = 10;
23     while(i > 0) {
24         if(!paused) {
25             printf("Counting: %d\n", i);
26             i--;
27         }
28         sleep(1);
29     }
30     printf("Countdown complete.\n");
31     return 0;
32 }
```

muzammil@muzammil: ~/Desktop/OS ...

```
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
./Q5_k232001
Countdown: 10
Countdown: 9
Countdown: 8
Countdown: 7
Countdown: 6
Countdown: 5
^C
Timer paused.
^Z
Timer resumed.
Countdown: 4
Countdown: 3
^C
Timer paused.
^Z
Timer resumed.
Countdown: 2
Countdown: 1
Countdown complete.
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Q6:

The screenshot shows a Linux desktop environment with a dark theme. At the top, a menu bar includes 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. Below the menu bar, a status bar displays 'May 4 22:05'. The main workspace is divided into three panes. The left pane shows a file explorer with icons for home, applications, and search. The middle pane displays a code editor with a file named 'Q5_k232001.c'. The code is as follows:

```
1 // k232001 - Muzammil
2 #include<stdio.h>
3 #include<signal.h>
4 #include<unistd.h>
5
6 volatile sig_atomic_t flag = 0;
7
8 void handler(int signum){
9     flag = !flag;
10 }
11
12 int main(){
13     signal(SIGUSR1, handler);
14     while(1){
15         if(flag)
16             printf("paused by SIGUSR1\n");
17         else
18             printf("Working...\n");
19         sleep(3);
20     }
21 }
```

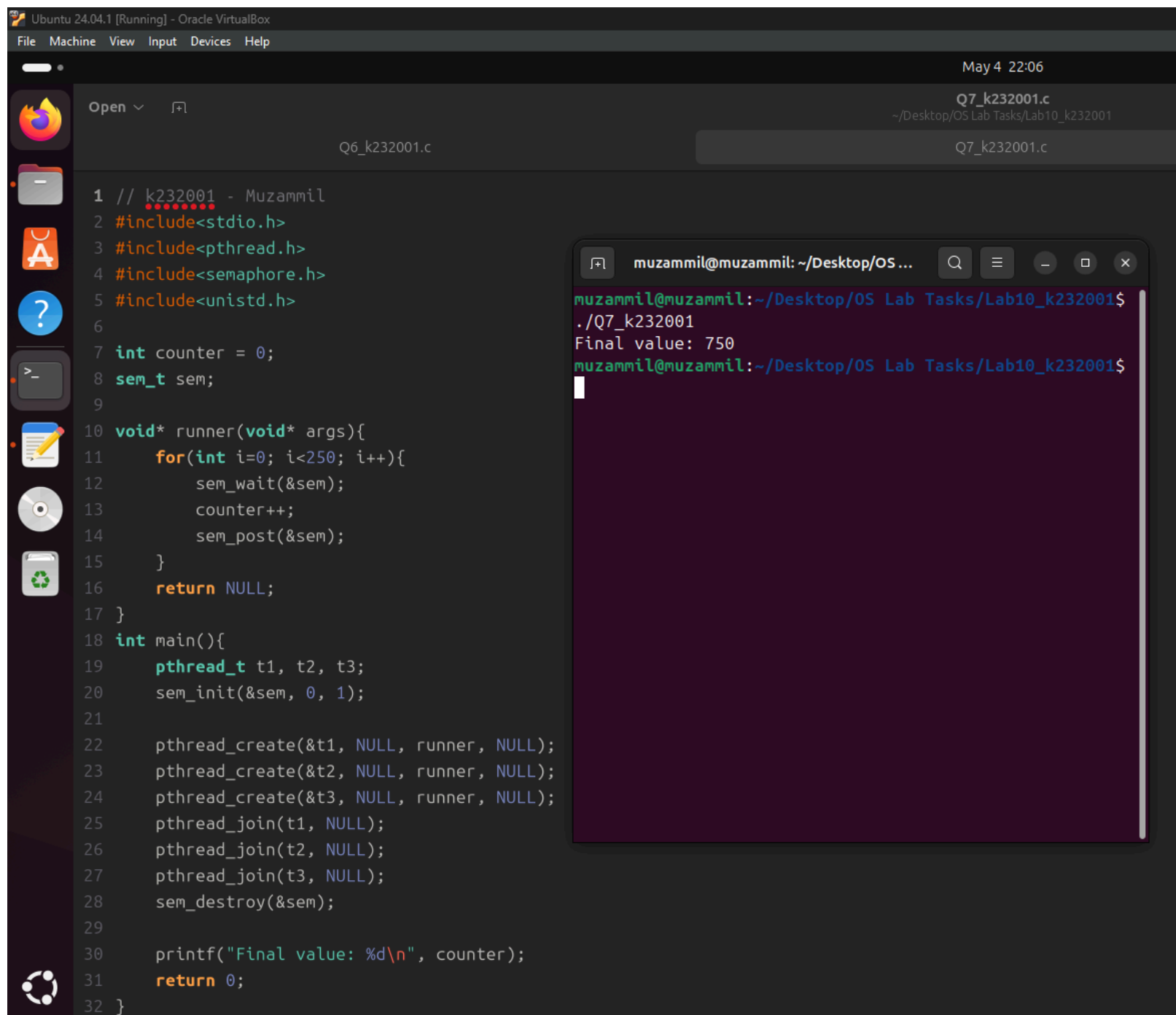
The right pane contains two terminal windows. The top terminal window shows the execution of the program 'Q6_k232001'. The output is as follows:

```
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$ ./Q6_k232001
Working...
Working...
Working...
Working...
Working...
paused by SIGUSR1
paused by SIGUSR1
paused by SIGUSR1
Working...
Working...
Working...
^C
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$
```

The bottom terminal window shows the execution of the 'pgrep' command to find the process ID of 'Q6_k232001'. The output is as follows:

```
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$ pgrep Q6_k232001
11942
14099
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$ kill -SIGUSR1 14099
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$ kill -SIGUSR1 14099
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Q7:



The image shows a screenshot of a code editor and a terminal window within a virtual machine environment. The code editor is displaying a C program named `Q6_k232001.c`. The program uses pthreads and semaphores to create three threads that increment a counter. The terminal window shows the execution of the program, displaying the final value of the counter as 750.

Ubuntu 24.04.1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

May 4 22:06

Q7_k232001.c
~/Desktop/OS Lab Tasks/Lab10_k232001

Q6_k232001.c

```
1 // k232001 - Muzammil
2 #include<stdio.h>
3 #include<pthread.h>
4 #include<semaphore.h>
5 #include<unistd.h>
6
7 int counter = 0;
8 sem_t sem;
9
10 void* runner(void* args){
11     for(int i=0; i<250; i++){
12         sem_wait(&sem);
13         counter++;
14         sem_post(&sem);
15     }
16     return NULL;
17 }
18 int main(){
19     pthread_t t1, t2, t3;
20     sem_init(&sem, 0, 1);
21
22     pthread_create(&t1, NULL, runner, NULL);
23     pthread_create(&t2, NULL, runner, NULL);
24     pthread_create(&t3, NULL, runner, NULL);
25     pthread_join(t1, NULL);
26     pthread_join(t2, NULL);
27     pthread_join(t3, NULL);
28     sem_destroy(&sem);
29
30     printf("Final value: %d\n", counter);
31     return 0;
32 }
```

muzammil@muzammil: ~/Desktop/OS ...

```
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
./Q7_k232001
Final value: 750
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
```


Q8:

Ubuntu 24.04.1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

May 4 22:08

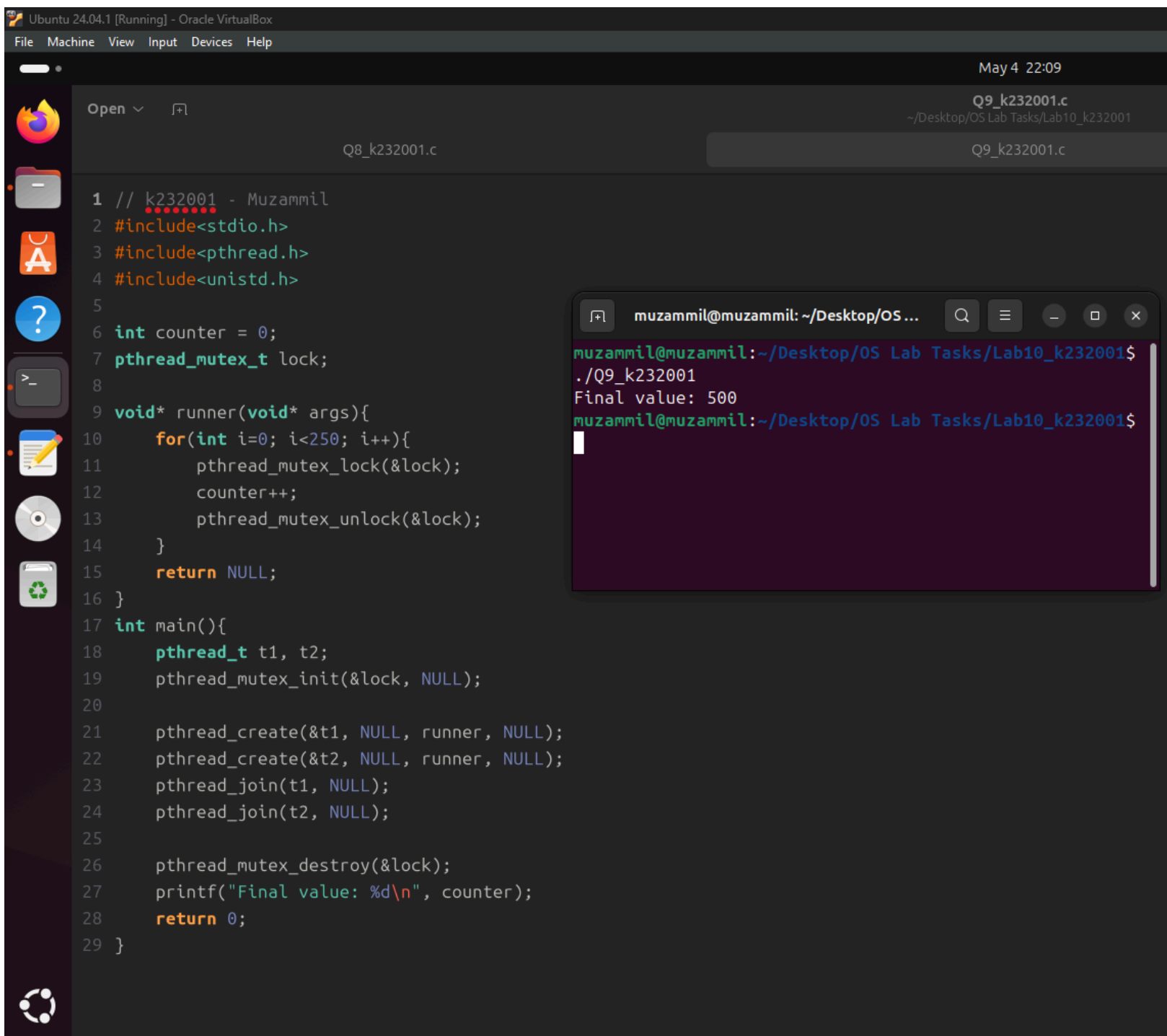
Q8_k232001.c

```
8 sem_t forks[N];
9 sem_t mutex;
10
11 void* philosopher(void* num){
12     int id = *(int*)num;
13     while(1) {
14         printf("Philosopher #%d is thinking\n", id);
15         sleep(1);
16         sem_wait(&mutex);
17         sem_wait(&forks[id]);
18         sem_wait(&forks[(id + 1) % N]);
19         sem_post(&mutex);
20
21         printf("Philosopher #%d is eating\n", id);
22         sleep(2);
23         sem_post(&forks[id]);
24         sem_post(&forks[(id + 1) % N]);
25     }
26     return NULL;
27 }
28 int main(){
29     pthread_t threads[N];
30     int ids[N];
31
32     sem_init(&mutex, 0, 1);
33     for(int i=0; i<N; i++){
34         sem_init(&forks[i], 0, 1);
35
36         for(int i=0; i<N; i++){
37             ids[i] = i;
38             pthread_create(&threads[i], NULL, philosopher, &ids[i]);
39         }
40         for(int i=0; i<N; i++){
41             pthread_join(threads[i], NULL);
```

muzammil@muzammil: ~/Desktop/OS ...

```
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
./Q8_k232001
Philosopher #0 is thinking
Philosopher #1 is thinking
Philosopher #2 is thinking
Philosopher #4 is thinking
Philosopher #3 is thinking
Philosopher #0 is eating
Philosopher #0 is thinking
Philosopher #1 is eating
Philosopher #4 is eating
Philosopher #1 is thinking
Philosopher #4 is thinking
Philosopher #3 is eating
Philosopher #3 is thinking
Philosopher #2 is eating
Philosopher #0 is eating
Philosopher #2 is thinking
Philosopher #0 is thinking
Philosopher #4 is eating
Philosopher #1 is eating
Philosopher #4 is thinking
Philosopher #1 is thinking
Philosopher #0 is eating
Philosopher #3 is eating
^C
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Q9:



Ubuntu 24.04.1 [Running] - Oracle VirtualBox

File Machine View Input Devices Help

May 4 22:09

Q9_k232001.c

~/Desktop/OS Lab Tasks/Lab10_k232001

Q9_k232001.c

```
1 // k232001 - Muzamil
2 #include<stdio.h>
3 #include<pthread.h>
4 #include<unistd.h>
5
6 int counter = 0;
7 pthread_mutex_t lock;
8
9 void* runner(void* args){
10     for(int i=0; i<250; i++){
11         pthread_mutex_lock(&lock);
12         counter++;
13         pthread_mutex_unlock(&lock);
14     }
15     return NULL;
16 }
17 int main(){
18     pthread_t t1, t2;
19     pthread_mutex_init(&lock, NULL);
20
21     pthread_create(&t1, NULL, runner, NULL);
22     pthread_create(&t2, NULL, runner, NULL);
23     pthread_join(t1, NULL);
24     pthread_join(t2, NULL);
25
26     pthread_mutex_destroy(&lock);
27     printf("Final value: %d\n", counter);
28     return 0;
29 }
```

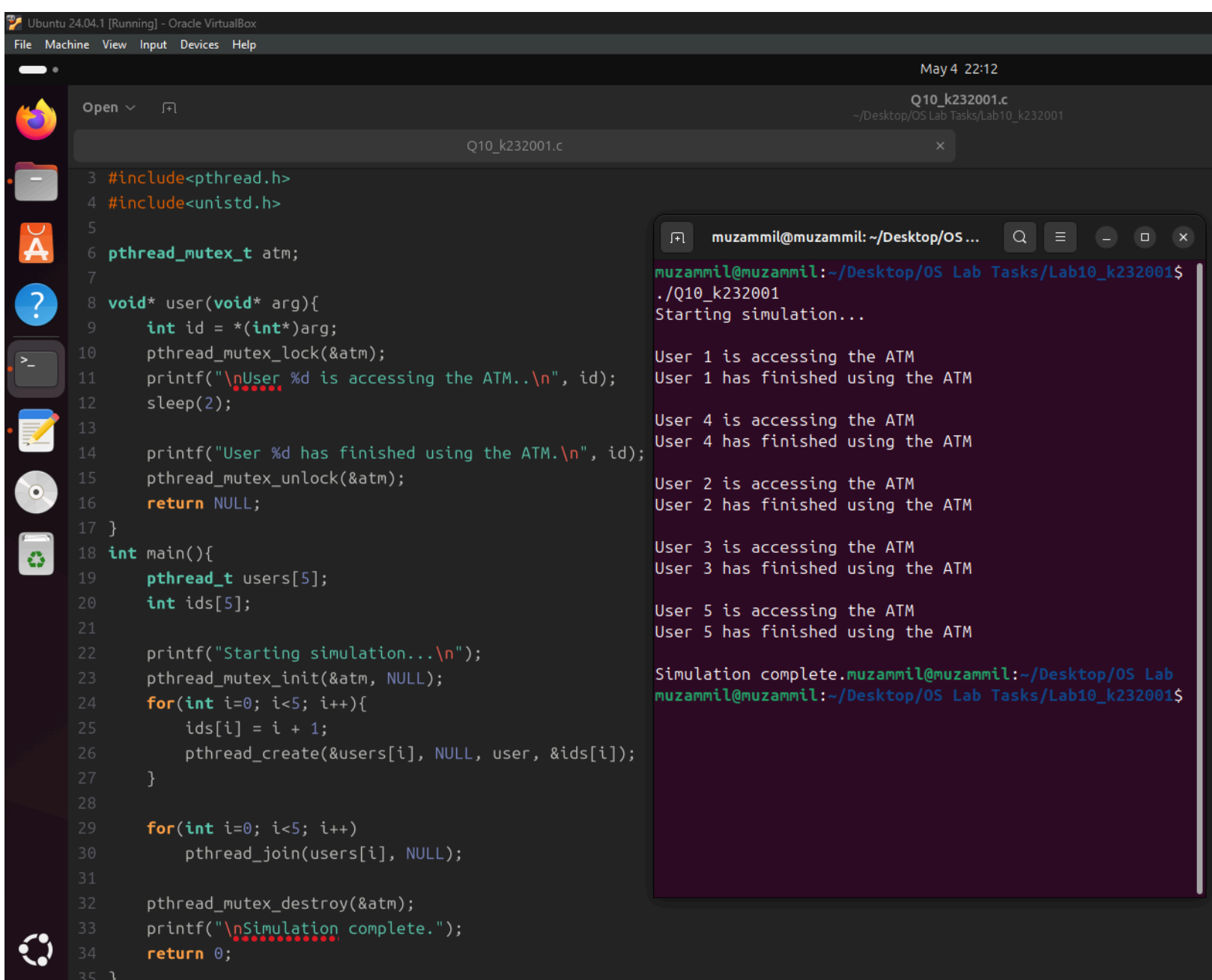
muzammil@muzammil: ~/Desktop/OS ...

muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001\$./Q9_k232001

Final value: 500

muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001\$

Q10:



The image shows a code editor window titled 'Q10_k232001.c' with a dark theme. The code is a C program for a multi-user ATM simulation using pthreads. It includes headers for pthread.h and unistd.h, defines a mutex 'atm', and implements a 'user' function that simulates a user accessing the ATM, sleeping for 2 seconds, and then finishing. The 'main' function creates 5 users, starts them, and waits for them to finish before printing 'Simulation complete.' and returning 0.

```
3 #include<pthread.h>
4 #include<unistd.h>
5
6 pthread_mutex_t atm;
7
8 void* user(void* arg){
9     int id = *(int*)arg;
10    pthread_mutex_lock(&atm);
11    printf("\nUser %d is accessing the ATM..\n", id);
12    sleep(2);
13
14    printf("User %d has finished using the ATM.\n", id);
15    pthread_mutex_unlock(&atm);
16    return NULL;
17 }
18 int main(){
19     pthread_t users[5];
20     int ids[5];
21
22     printf("Starting simulation...\n");
23     pthread_mutex_init(&atm, NULL);
24     for(int i=0; i<5; i++){
25         ids[i] = i + 1;
26         pthread_create(&users[i], NULL, user, &ids[i]);
27     }
28
29     for(int i=0; i<5; i++)
30         pthread_join(users[i], NULL);
31
32     pthread_mutex_destroy(&atm);
33     printf("\nSimulation complete.");
34     return 0;
35 }
```

On the right, a terminal window shows the execution of the program. It displays the output of the simulation, including the sequence of users accessing the ATM and the final completion message.

```
muzammil@muzammil: ~/Desktop/OS Lab Tasks/Lab10_k232001$
./Q10_k232001
Starting simulation...

User 1 is accessing the ATM
User 1 has finished using the ATM

User 4 is accessing the ATM
User 4 has finished using the ATM

User 2 is accessing the ATM
User 2 has finished using the ATM

User 3 is accessing the ATM
User 3 has finished using the ATM

User 5 is accessing the ATM
User 5 has finished using the ATM

Simulation complete.muzammil@muzammil:~/Desktop/OS Lab
muzammil@muzammil:~/Desktop/OS Lab Tasks/Lab10_k232001$
```

Scheduling Algorithm Tasks

Task01:

23K-2001

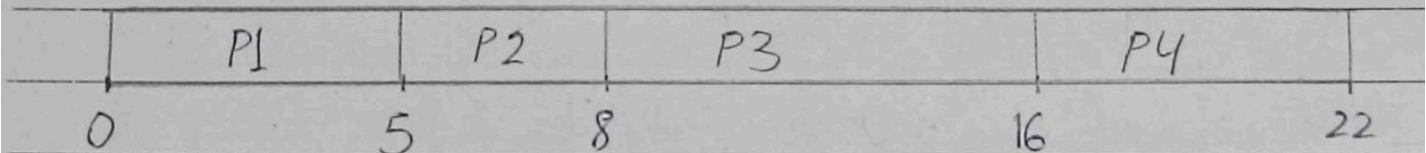
OS Lab10: Scheduling Algorithm Tasks

Date: _____

Task 01: FCFS

Process #	Arrival time	Burst time	Start time	Completion time	Turnaround time	Waiting time
P1	0	5	0	5	5	0
P2	1	3	5	8	7	4
P3	2	8	8	16	14	6
P4	3	6	16	22	19	13

Gantt Chart:



$$\text{Avg. turnaround time} \Rightarrow (5 + 7 + 14 + 19) / 4 = 11.25$$

$$\text{Avg. waiting time} \Rightarrow (0 + 4 + 6 + 13) / 4 = 5.75$$

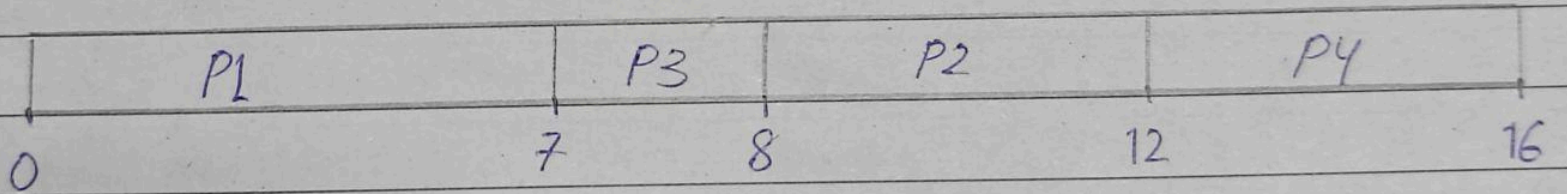
Task02:

Date: _____

Task 02: SJF - Non preemptive

Process #	Arrival time	Burst time	Start time	Completion time	Turnaround time	Waiting time
P1	0	7	0	7	7	0
P2	2	4	8	12	10	6
P3	4	1	7	8	4	3
P4	5	4	12	16	11	7

Gantt Chart:



$$\text{Avg. turnaround time} \Rightarrow (7+10+4+11)/4 = 8$$

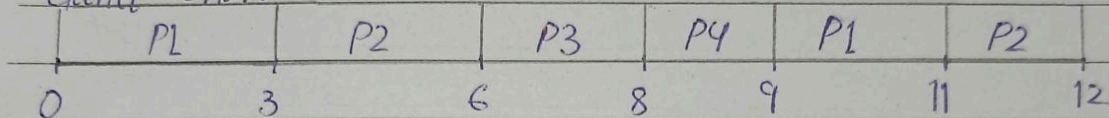
$$\text{Avg. waiting time} \Rightarrow (0+6+3+7)/4 = 4$$

Task03 & Task04:

Task03: Round Robin

Process#	Arrival time	Burst time	Completion time	Turnaround time	Waiting time
P1	0	5	11	11	6
P2	1	4	12	11	7
P3	2	2	8	6	4
P4	3	1	9	6	5

Gantt Chart:



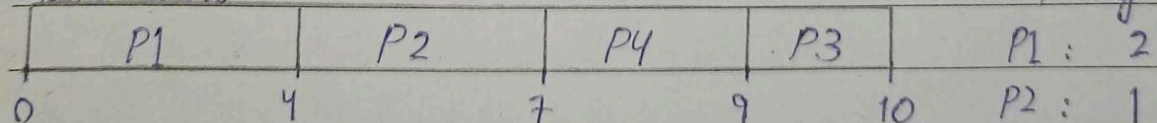
$$\text{Avg. turnaround time} \Rightarrow (11 + 11 + 6 + 6) / 4 = 8.5$$

$$\text{Avg. waiting time} \Rightarrow (6 + 7 + 4 + 5) / 4 = 5.5$$

Task04: Priority - Non preemptive

Process#	Arrival time	Burst time	Start time	Completion time	Turnaround time	Waiting time
P1	0	4	0	4	4	0
P2	1	3	4	7	6	3
P3	2	1	9	10	8	7
P4	3	2	7	9	6	4

Gantt Chart:



Priority:

P1 : 2

P2 : 1

P3 : 4

P4 : 3

$$\text{Avg. turnaround time} \Rightarrow (4 + 6 + 8 + 6) = 6$$

$$\text{Avg. waiting time} \Rightarrow (0 + 3 + 7 + 4) = 3.5$$