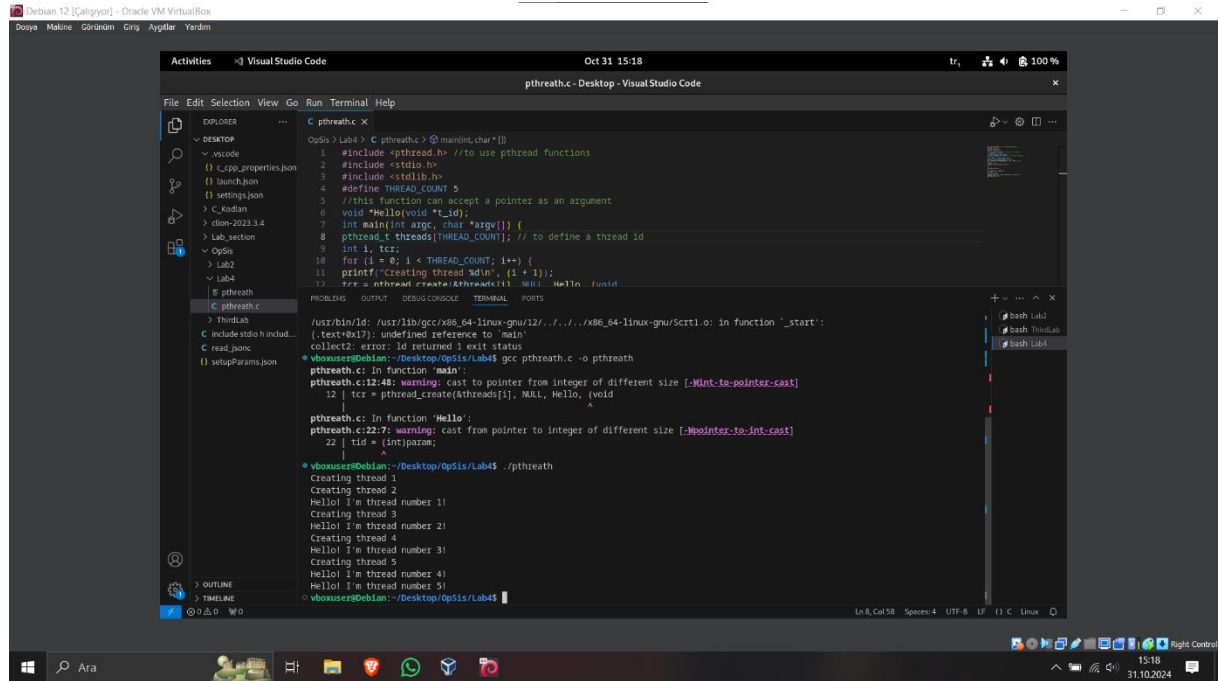


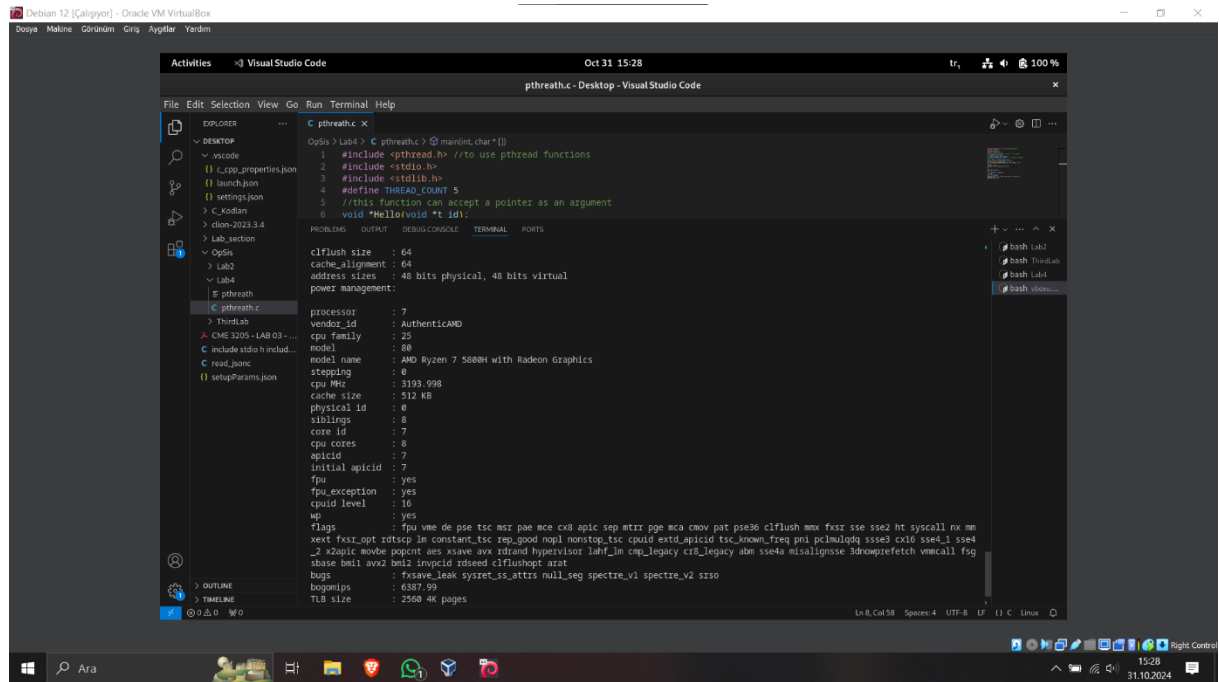
# Çağrı AYDIN 2021510010 – Lab 3

## Code 1:



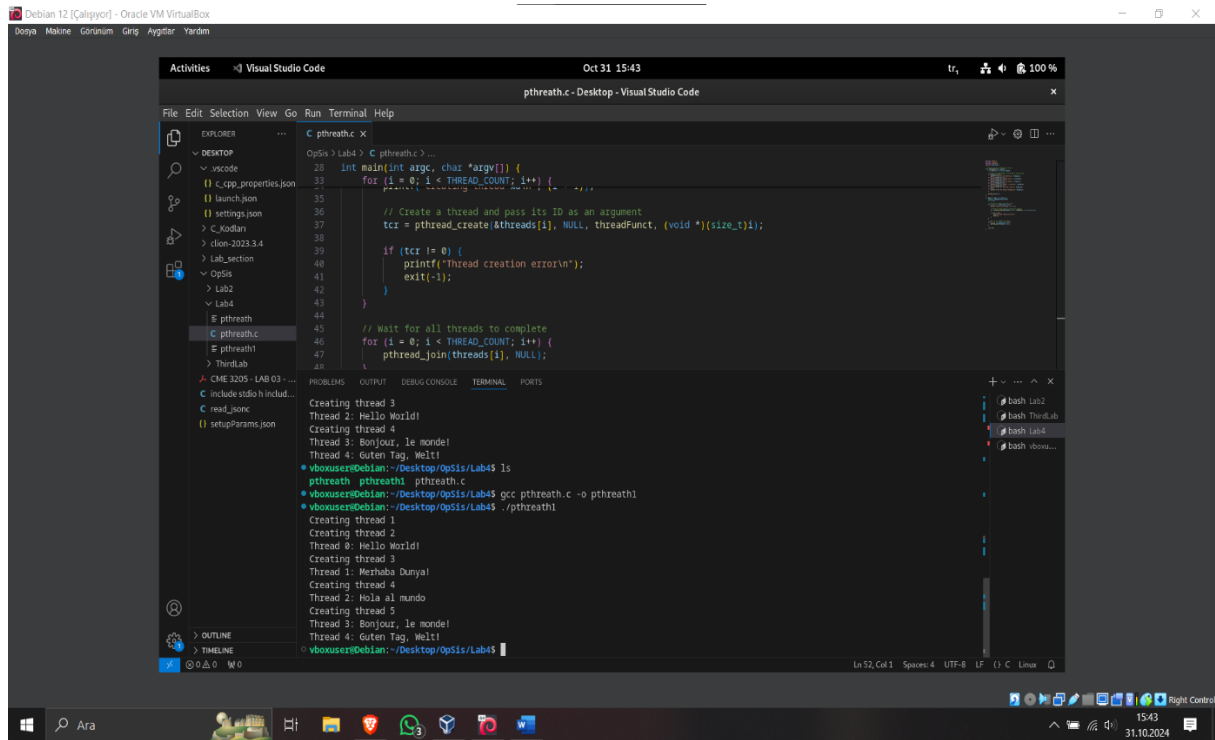
```
1 #include <pthread.h> //to use pthread functions
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define THREAD_COUNT 5
5 //this function can accept a pointer as an argument
6 void *Hello(void *t_id);
7 int main(int argc, char *argv[]) {
8     pthread_t threads[THREAD_COUNT]; // to define a thread id
9     int i, tcr;
10    for (i = 0; i < THREAD_COUNT; i++) {
11        printf("creating thread %d\n", (i + 1));
12        tcr = pthread_create(&threads[i], NULL, Hello, (void *)
13        pthread_exit(&threads[i], NULL, Hello, (void *)
14        pthread_exit(&threads[i], NULL, Hello, (void *)
15        pthread_exit(&threads[i], NULL, Hello, (void *)
16        pthread_exit(&threads[i], NULL, Hello, (void *)
17        pthread_exit(&threads[i], NULL, Hello, (void *)
18        pthread_exit(&threads[i], NULL, Hello, (void *)
19        pthread_exit(&threads[i], NULL, Hello, (void *)
20        pthread_exit(&threads[i], NULL, Hello, (void *)
21        pthread_exit(&threads[i], NULL, Hello, (void *)
22        pthread_exit(&threads[i], NULL, Hello, (void *)
23        pthread_exit(&threads[i], NULL, Hello, (void *)
24        pthread_exit(&threads[i], NULL, Hello, (void *)
25        pthread_exit(&threads[i], NULL, Hello, (void *)
26        pthread_exit(&threads[i], NULL, Hello, (void *)
27        pthread_exit(&threads[i], NULL, Hello, (void *)
28        pthread_exit(&threads[i], NULL, Hello, (void *)
29        pthread_exit(&threads[i], NULL, Hello, (void *)
30        pthread_exit(&threads[i], NULL, Hello, (void *)
31        pthread_exit(&threads[i], NULL, Hello, (void *)
32        pthread_exit(&threads[i], NULL, Hello, (void *)
33        pthread_exit(&threads[i], NULL, Hello, (void *)
34        pthread_exit(&threads[i], NULL, Hello, (void *)
35        pthread_exit(&threads[i], NULL, Hello, (void *)
36        pthread_exit(&threads[i], NULL, Hello, (void *)
37        pthread_exit(&threads[i], NULL, Hello, (void *)
38        pthread_exit(&threads[i], NULL, Hello, (void *)
39        pthread_exit(&threads[i], NULL, Hello, (void *)
40        pthread_exit(&threads[i], NULL, Hello, (void *)
41        pthread_exit(&threads[i], NULL, Hello, (void *)
42        pthread_exit(&threads[i], NULL, Hello, (void *)
43        pthread_exit(&threads[i], NULL, Hello, (void *)
44        pthread_exit(&threads[i], NULL, Hello, (void *)
45        pthread_exit(&threads[i], NULL, Hello, (void *)
46        pthread_exit(&threads[i], NULL, Hello, (void *)
47        pthread_exit(&threads[i], NULL, Hello, (void *)
48        pthread_exit(&threads[i], NULL, Hello, (void *)
49        pthread_exit(&threads[i], NULL, Hello, (void *)
50        pthread_exit(&threads[i], NULL, Hello, (void *)
51        pthread_exit(&threads[i], NULL, Hello, (void *)
52        pthread_exit(&threads[i], NULL, Hello, (void *)
53        pthread_exit(&threads[i], NULL, Hello, (void *)
54        pthread_exit(&threads[i], NULL, Hello, (void *)
55        pthread_exit(&threads[i], NULL, Hello, (void *)
56        pthread_exit(&threads[i], NULL, Hello, (void *)
57        pthread_exit(&threads[i], NULL, Hello, (void *)
58        pthread_exit(&threads[i], NULL, Hello, (void *)
59        pthread_exit(&threads[i], NULL, Hello, (void *)
60        pthread_exit(&threads[i], NULL, Hello, (void *)
61        pthread_exit(&threads[i], NULL, Hello, (void *)
62        pthread_exit(&threads[i], NULL, Hello, (void *)
63        pthread_exit(&threads[i], NULL, Hello, (void *)
64        pthread_exit(&threads[i], NULL, Hello, (void *)
65        pthread_exit(&threads[i], NULL, Hello, (void *)
66        pthread_exit(&threads[i], NULL, Hello, (void *)
67        pthread_exit(&threads[i], NULL, Hello, (void *)
68        pthread_exit(&threads[i], NULL, Hello, (void *)
69        pthread_exit(&threads[i], NULL, Hello, (void *)
70        pthread_exit(&threads[i], NULL, Hello, (void *)
71        pthread_exit(&threads[i], NULL, Hello, (void *)
72        pthread_exit(&threads[i], NULL, Hello, (void *)
73        pthread_exit(&threads[i], NULL, Hello, (void *)
74        pthread_exit(&threads[i], NULL, Hello, (void *)
75        pthread_exit(&threads[i], NULL, Hello, (void *)
76        pthread_exit(&threads[i], NULL, Hello, (void *)
77        pthread_exit(&threads[i], NULL, Hello, (void *)
78        pthread_exit(&threads[i], NULL, Hello, (void *)
79        pthread_exit(&threads[i], NULL, Hello, (void *)
80        pthread_exit(&threads[i], NULL, Hello, (void *)
81        pthread_exit(&threads[i], NULL, Hello, (void *)
82        pthread_exit(&threads[i], NULL, Hello, (void *)
83        pthread_exit(&threads[i], NULL, Hello, (void *)
84        pthread_exit(&threads[i], NULL, Hello, (void *)
85        pthread_exit(&threads[i], NULL, Hello, (void *)
86        pthread_exit(&threads[i], NULL, Hello, (void *)
87        pthread_exit(&threads[i], NULL, Hello, (void *)
88        pthread_exit(&threads[i], NULL, Hello, (void *)
89        pthread_exit(&threads[i], NULL, Hello, (void *)
90        pthread_exit(&threads[i], NULL, Hello, (void *)
91        pthread_exit(&threads[i], NULL, Hello, (void *)
92        pthread_exit(&threads[i], NULL, Hello, (void *)
93        pthread_exit(&threads[i], NULL, Hello, (void *)
94        pthread_exit(&threads[i], NULL, Hello, (void *)
95        pthread_exit(&threads[i], NULL, Hello, (void *)
96        pthread_exit(&threads[i], NULL, Hello, (void *)
97        pthread_exit(&threads[i], NULL, Hello, (void *)
98        pthread_exit(&threads[i], NULL, Hello, (void *)
99        pthread_exit(&threads[i], NULL, Hello, (void *)
100       pthread_exit(&threads[i], NULL, Hello, (void *)
```

## My Processor:



```
processor       : 7
vendor_id     : AuthenticAMD
cpu family    : 25
model        : 80
model name    : AMD Ryzen 7 5800H with Radeon Graphics
stepping     : 0
cpu MHz      : 3193.998
cache size    : 512 KB
physical id   : 0
siblings     : 8
core id      : 7
cpu cores    : 8
apicid       : 7
initial apicid : 7
fpu          : yes
fpu_exception : yes
cpuid level  : 16
wp           : yes
flags        : fpu vme de pse tsc mtr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mm
xext fxsr_opt rdrscp lm constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid tsc_known_freq pni pclmulqdq sse3 cx16 sse4_1 sse4_2
x2apic movbe popcnt aes xsave avx xsaveopt xsavecvt rdtscp arat
bugs          : fsave_leak sysret_ss_attrs null_seg spectre_v1 spectre_v2 srsio
bogomips     : 6387.99
tlb size     : 2560 4K pages
```

# Task 1:



The screenshot shows a Visual Studio Code editor window titled "pthread.c - Desktop - Visual Studio Code". The editor is running on a Debian 12 VM. The file explorer on the left shows the project structure, including files like c\_cpp\_properties.json, launch.json, settings.json, and pthread.c. The main editor window displays the source code of pthread.c, which is a C program that creates and joins threads. The code is as follows:

```
28 int main(int argc, char *argv[]) {
29     for (i = 0; i < THREAD_COUNT; i++) {
30         pthread_create(&threads[i], NULL, threadFunc, (void *)(&i));
31     }
32     // Create a thread and pass its ID as an argument
33     tcr = pthread_create(&threads[i], NULL, threadFunc, (void *)(&i));
34     if (tcr != 0) {
35         printf("Thread creation error\n");
36         exit(-1);
37     }
38     // Wait for all threads to complete
39     for (i = 0; i < THREAD_COUNT; i++) {
40         pthread_join(threads[i], NULL);
41     }
42 }
```

The terminal output shows the execution of the program, which prints messages from the main thread and five child threads. The output is as follows:

```
Creating thread 3
Thread 2: Hello World!
Creating thread 4
Thread 3: Bonjour, le monde!
Thread 4: Guten Tag, Welt!
vboxuser@Debian:~/Desktop/OpSis/Lab4$ ls
pthread pthread1 pthread.c
vboxuser@Debian:~/Desktop/OpSis/Lab4$ gcc pthread.c -o pthread1
vboxuser@Debian:~/Desktop/OpSis/Lab4$ ./pthread1
Creating thread 1
Creating thread 2
Thread 0: Hello World!
Creating thread 3
Thread 1: Meshaba Dunya!
Creating thread 4
Thread 2: Hola al mundo
Creating thread 5
Thread 3: Bonjour, le monde!
Thread 4: Guten Tag, Welt!
```

```

#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#define THREAD_COUNT 5

void *threadFunc(void *param) {
    int thread_id = (int)(size_t)param;

    if (thread_id == 0) {
        printf("Thread %d: Hello World!\n", thread_id);
    } else if (thread_id == 1) {
        printf("Thread %d: Merhaba Dunya!\n", thread_id);
    } else if (thread_id == 2) {
        printf("Thread %d: Hola al mundo\n", thread_id);
    } else if (thread_id == 3) {
        printf("Thread %d: Bonjour, le monde!\n", thread_id);
    } else if (thread_id == 4) {
        printf("Thread %d: Guten Tag, Welt!\n", thread_id);
    } else {
        printf("Thread %d: Unknown language!\n", thread_id);
    }
    pthread_exit(NULL);
}

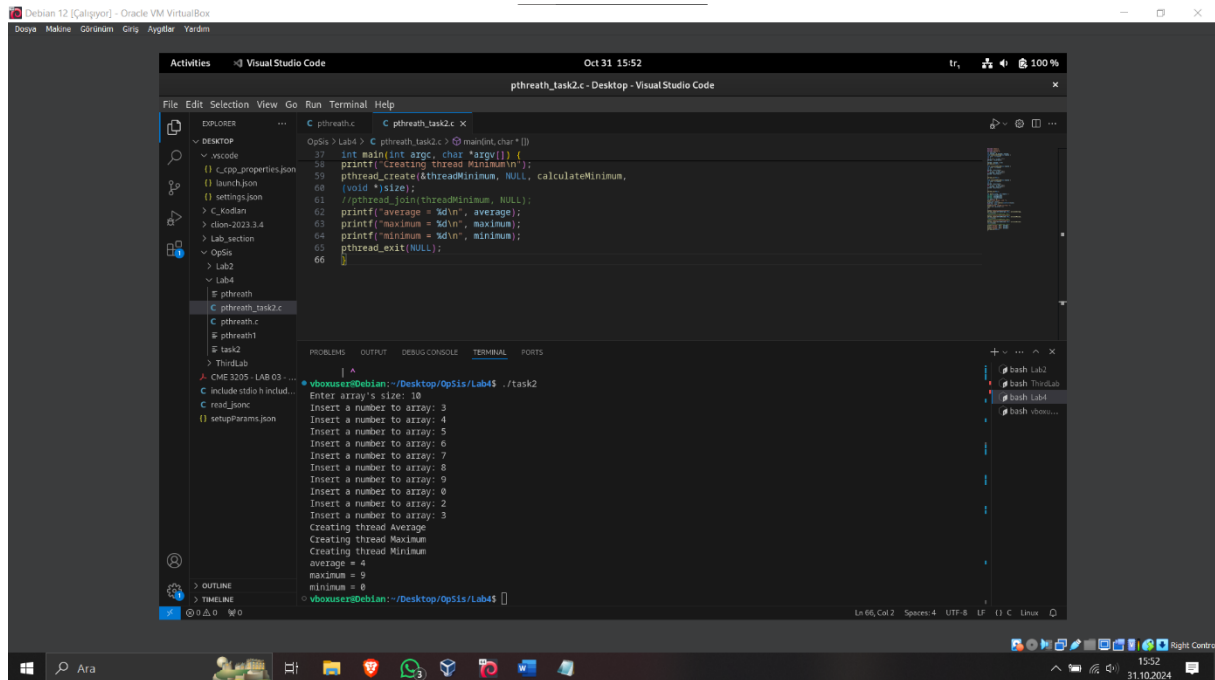
int main(int argc, char *argv[]) {
    pthread_t threads[THREAD_COUNT];
    int i, tcr;
    // Create each thread and assign a task
    for (i = 0; i < THREAD_COUNT; i++) {
        printf("Creating thread %d\n", (i + 1));

        tcr = pthread_create(&threads[i], NULL, threadFunc, (void *) (size_t)i);

        if (tcr != 0) {
            printf("Thread creation error\n");
            exit(-1);
        }
    }
    for (i = 0; i < THREAD_COUNT; i++) {
        pthread_join(threads[i], NULL);
    }
    return 0;
}

```

## Task 2:



```
OpSis > Lab4 > C pthread_task2.c > main(int, char *) {
37 int main(int argc, char *argv[]) {
38     printf("creating thread minimum\n");
39     pthread_create(&threadMinimum, NULL, calculateMinimum,
68     (void *)size);
61 //pthread_join(threadMinimum, NULL);
62 printf("average = %d\n", average);
63 printf("maximum = %d\n", maximum);
64 printf("minimum = %d\n", minimum);
65 pthread_exit(NULL);
66 }
```

```
Enter array's size: 10
Insert a number to array: 3
Insert a number to array: 4
Insert a number to array: 5
Insert a number to array: 6
Insert a number to array: 7
Insert a number to array: 8
Insert a number to array: 9
Insert a number to array: 0
Insert a number to array: 2
Insert a number to array: 3
Creating thread Average
Creating thread Maximum
Creating thread Minimum
average = 4
maximum = 9
minimum = 0
vbmusers@Debian:~/Desktop/OpSis/Lab4$ ./task2
```

**Explanation:** Yes, this code works parallel for calculating min, max and avg. number after getting the array size and array elements for time efficiency.

### Task 3:

In this part our threads trying to access same global variables at the same time, so it occurs a data race. Threads access at the same time and trying to write different values so basically, we have a synchronization issue. To fix this we can make a flag if a thread is already using that data or we can make a pipeline between threads.