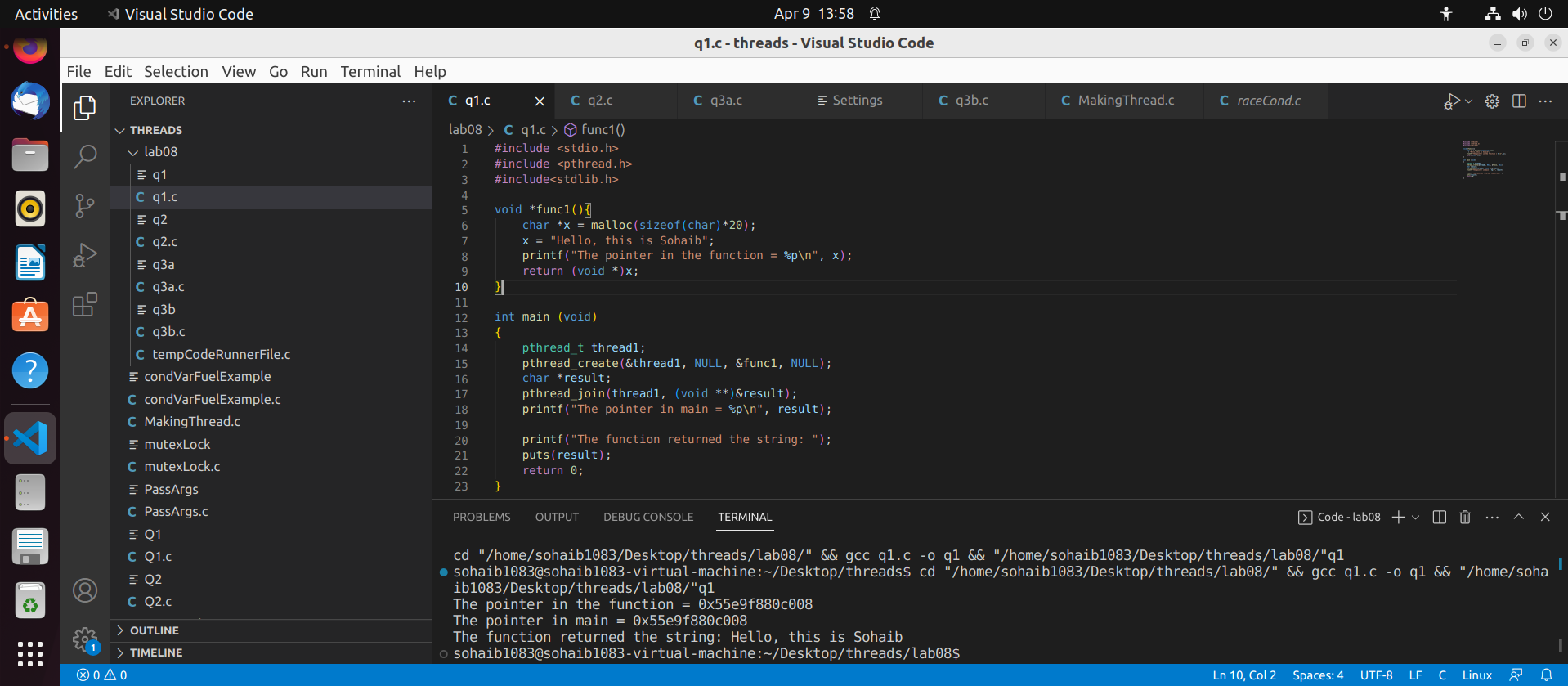
**LAB 08**

**21K-3278-D**

Q1.



#include <stdio.h>

#include <pthread.h>

#include<stdlib.h>

void \*func1(){

char \*x = malloc(sizeof(char)\*20);

x = "Hello, this is Sohaib";

printf("The pointer in the function = %p\n", x);

return (void \*)x;

}

int main (void)

{

pthread\_t thread1;

pthread\_create(&thread1, NULL, &func1, NULL);

char \*result;

pthread\_join(thread1, (void \*\*)&result);

printf("The pointer in main = %p\n", result);

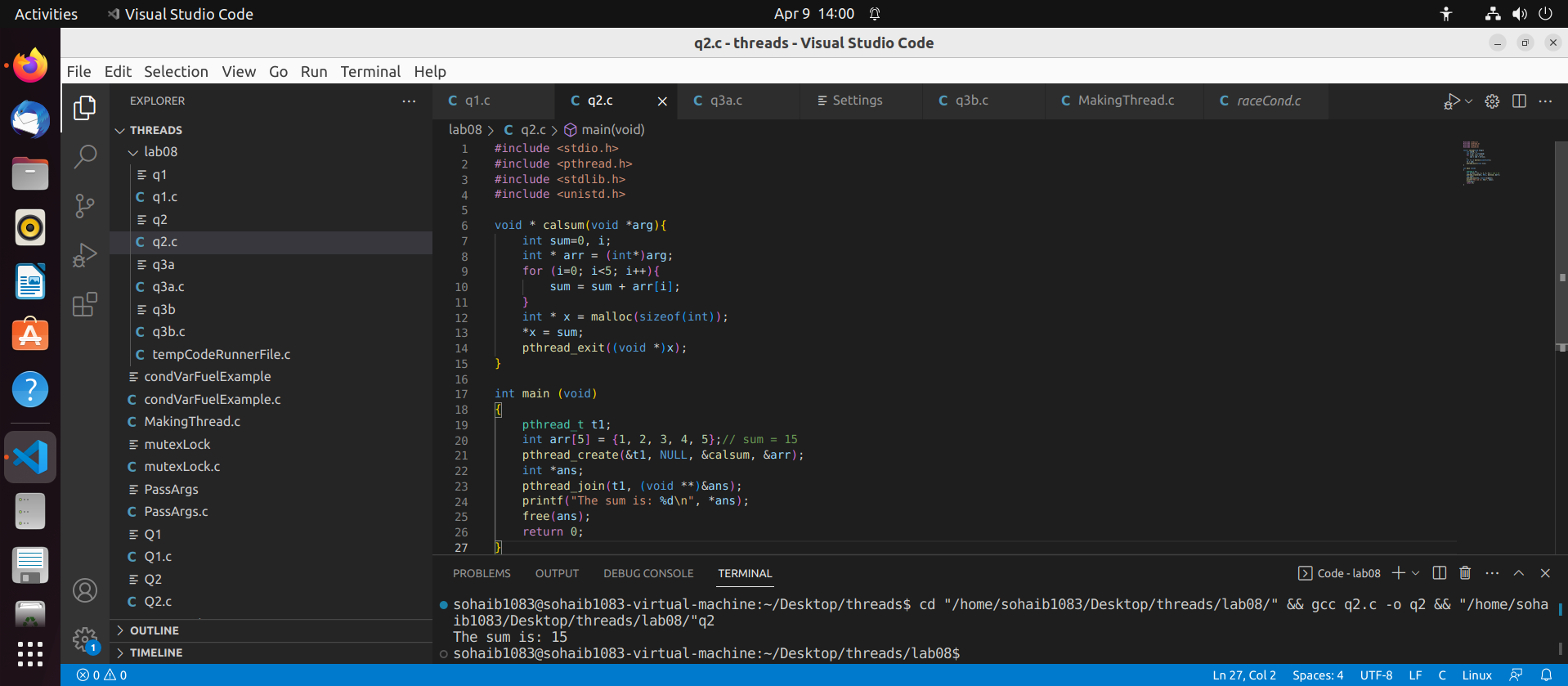
printf("The function returned the string: ");

puts(result);

return 0;

}

Q2.



#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <unistd.h>

void \* calsum(void \*arg){

int sum=0, i;

int \* arr = (int\*)arg;

for (i=0; i<5; i++){

sum = sum + arr[i];

}

int \* x = malloc(sizeof(int));

\*x = sum;

pthread\_exit((void \*)x);

}

int main (void)

{

pthread\_t t1;

int arr[5] = {1, 2, 3, 4, 5};// sum = 15

pthread\_create(&t1, NULL, &calsum, &arr);

int \*ans;

pthread\_join(t1, (void \*\*)&ans);

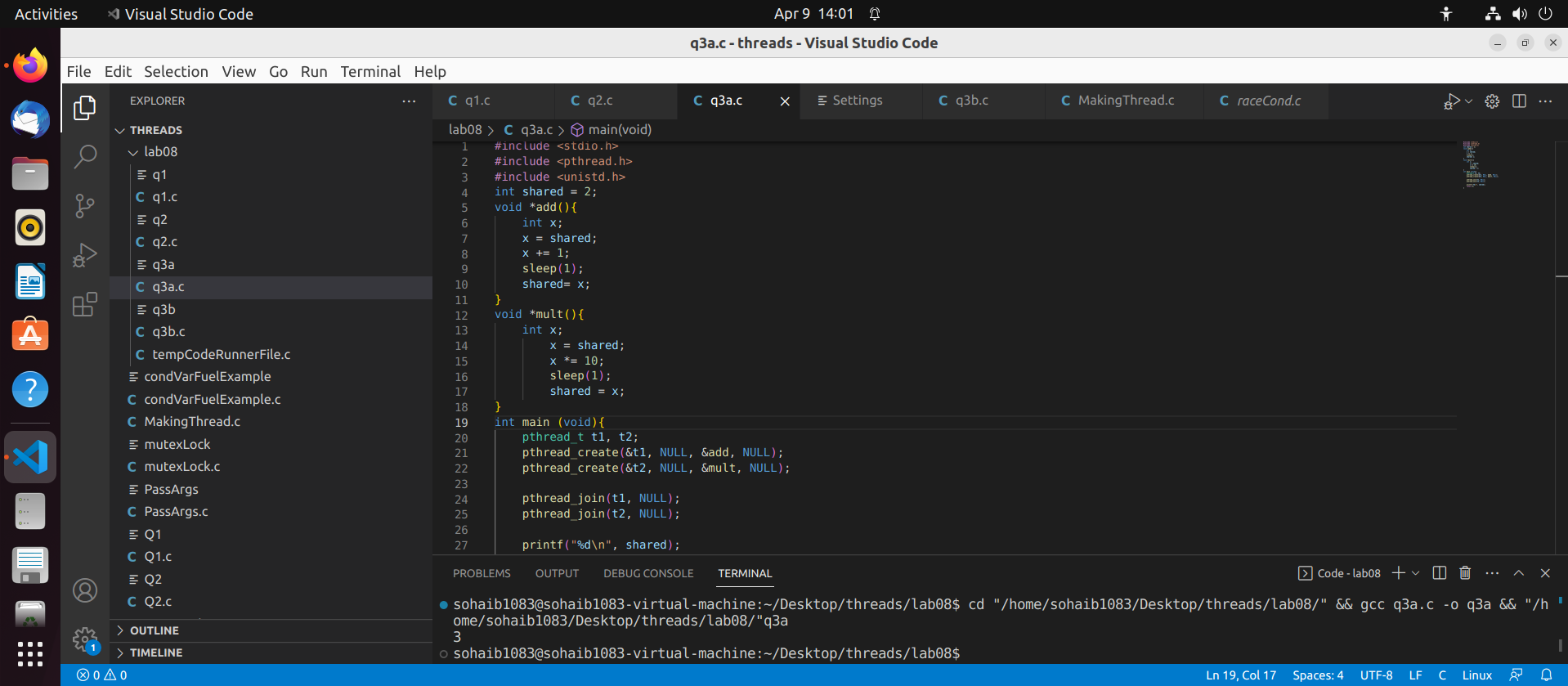
printf("The sum is: %d\n", \*ans);

free(ans);

return 0;

}

Q3 a.



#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

int shared = 2;

void \*add(){

int x;

x = shared;

x += 1;

sleep(1);

shared= x;

}

void \*mult(){

int x;

x = shared;

x \*= 10;

sleep(1);

shared = x;

}

int main (void){

pthread\_t t1, t2;

pthread\_create(&t1, NULL, &add, NULL);

pthread\_create(&t2, NULL, &mult, NULL);

pthread\_join(t1, NULL);

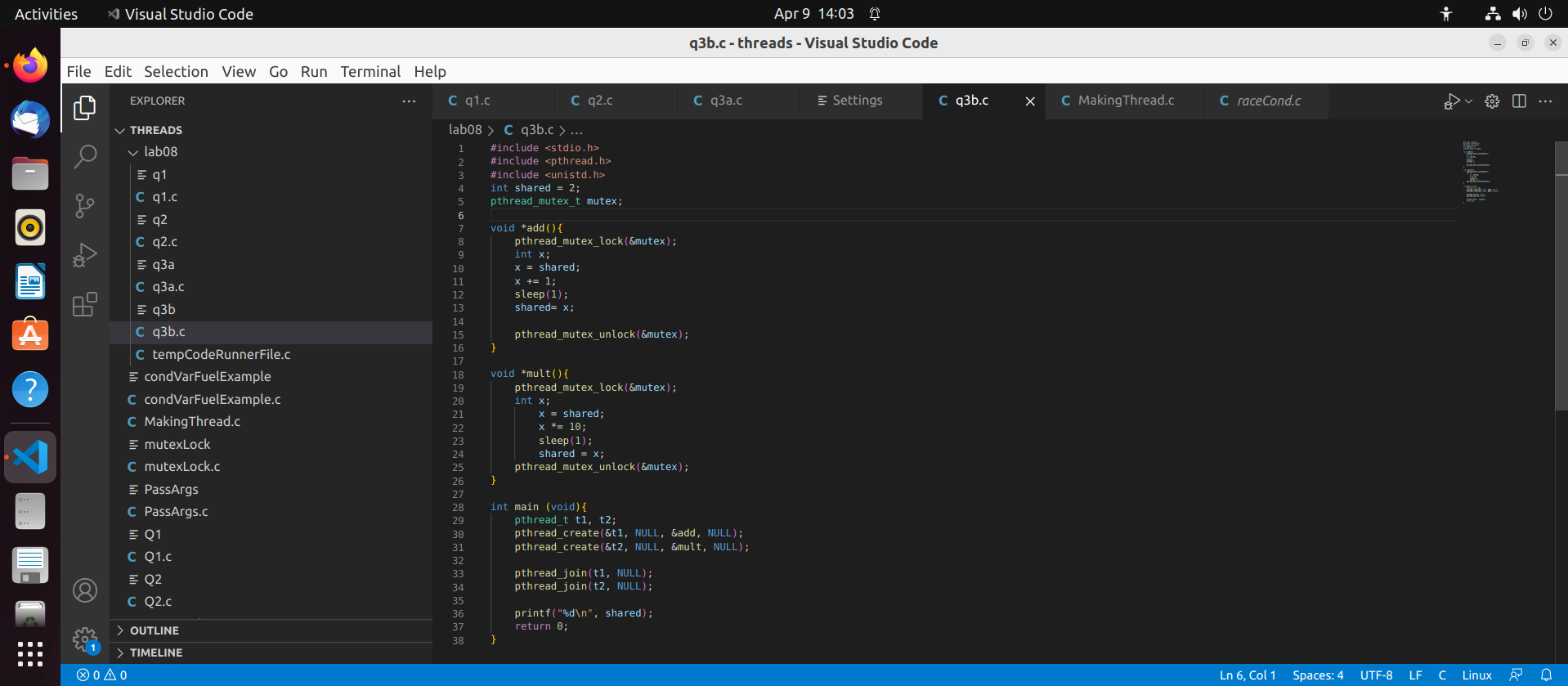
pthread\_join(t2, NULL);

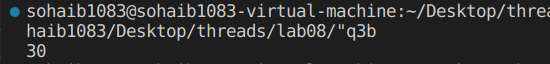
printf("%d\n", shared);

return 0;

}

Q3 b.





#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

int shared = 2;

pthread\_mutex\_t mutex;

void \*add(){

pthread\_mutex\_lock(&mutex);

int x;

x = shared;

x += 1;

sleep(1);

shared= x;

pthread\_mutex\_unlock(&mutex);

}

void \*mult(){

pthread\_mutex\_lock(&mutex);

int x;

x = shared;

x \*= 10;

sleep(1);

shared = x;

pthread\_mutex\_unlock(&mutex);

}

int main (void){

pthread\_t t1, t2;

pthread\_create(&t1, NULL, &add, NULL);

pthread\_create(&t2, NULL, &mult, NULL);

pthread\_join(t1, NULL);

pthread\_join(t2, NULL);

printf("%d\n", shared);

return 0;

}

Q3.c

In part a where there was a race condition between t1 and t2, two outputs were possible (3 and 21) which were both incorrect due the preemption(sleep();) happening in the process of updating shared value. After implementing mutex in part b, only one thread enters the critical section at a time producing only one result.