OPERATING SYSTEM LABORATORY WORKSHEET

NAME: RUDRA KANIYA	**REGISTRATION NUMBER: 11803187**
ROLLNO: B-40 (K18MS)	**QUESTION NO: 8**

Question 8 a). Write a program to implement the solution of dining philosopher problem.

GITHUB LINK: https://github.com/Rudrakaniya/OS_Lab/blob/master/diningProb.c

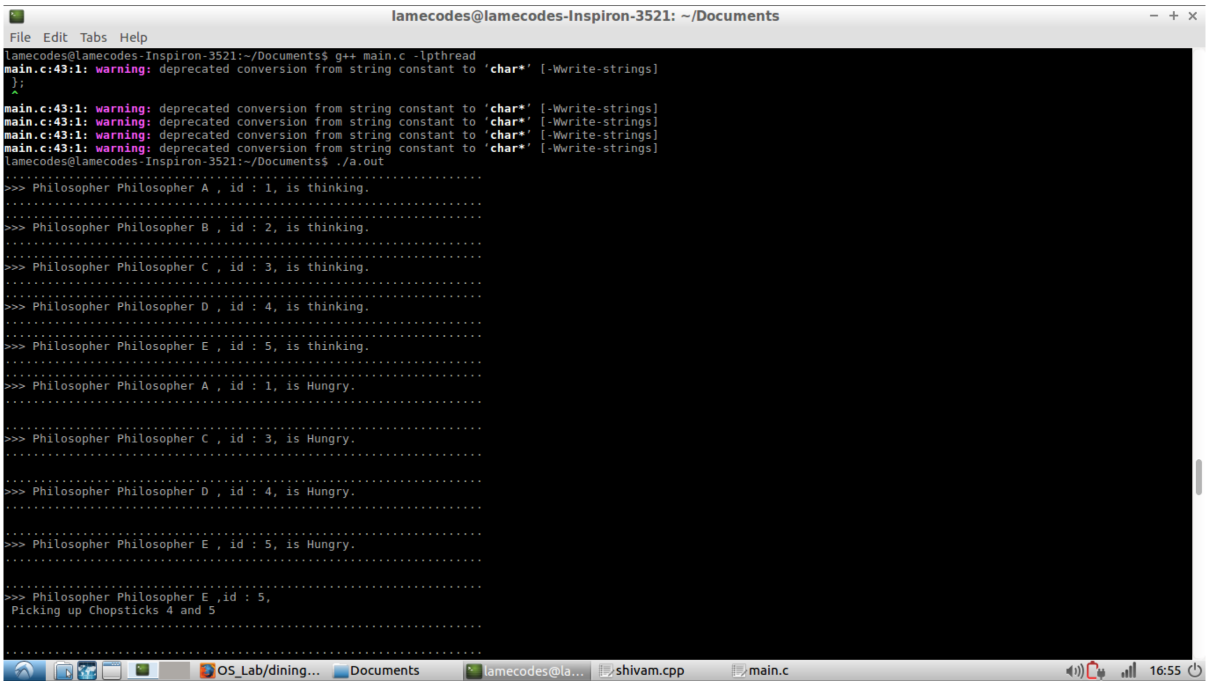
CODE:

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#define end "\n"
#define LEFT (PhilNum + 4) % 5
#define RIGHT (PhilNum + 1) % 5

//semaphore declaration
sem_t quantum;
sem_t boo[5];


//three states of philosophers
enum anvi{
 EATING , HUNGRY , THINKING
};

struct Philosopher{
    char * name;
    int id;
};

//giving the values to the struct
struct Philosopher P[5]={
    {
        "Philosopher A",0
    },
    {
        "Philosopher B",1
    },
    {
        "Philosopher C",2
    },
    {
        "Philosopher D",3
    },
    {
        "Philosopher E",4
    }
};

//philosopher flag = globle decleration for the current state of every professer.
int pflag[5];


void test(int PhilNum)
```

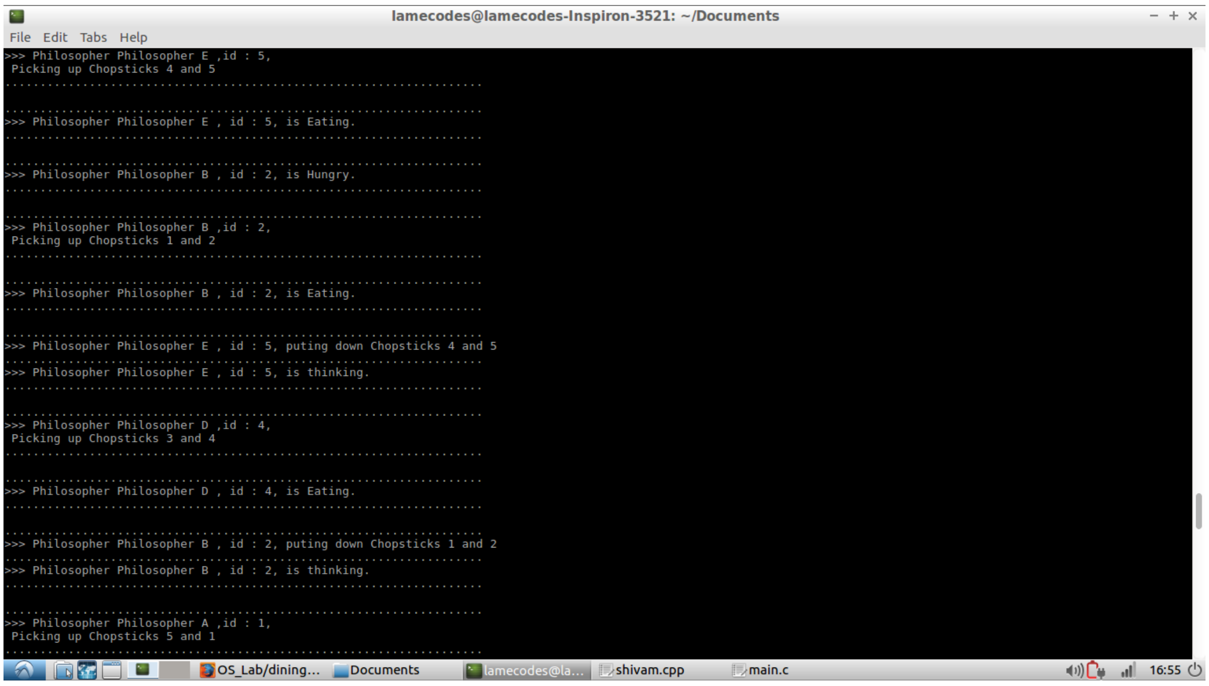
```c
{
    if ( pflag[LEFT] != EATING && pflag[RIGHT] != EATING){

        pflag[PhilNum] = EATING;

        sleep(2);
        printf("....................................................................\n");
        printf(">>> Philosopher %s ,id : %d,\n Picking up Chopsticks %d and %d  \n",P[PhilNum].name, PhilNum + 1, LEFT + 1, PhilNum + 1);
        printf("....................................................................\n\n");

        printf("....................................................................\n");
        printf(">>> Philosopher %s , id : %d, is Eating.\n",P[PhilNum].name, PhilNum + 1);
        printf("....................................................................\n\n");
        sem_post(&boo[PhilNum]);
    }
}

void take_chopsticks(int PhilNum)
{

    sem_wait(&quantum); /* critical section */

    pflag[PhilNum] = HUNGRY;

    printf("....................................................................\n");
    printf(">>> Philosopher %s , id : %d, is Hungry.\n",P[PhilNum].name, PhilNum + 1);
    printf("....................................................................\n\n");

    test(PhilNum);
    sem_post(&quantum); /* end critical section */

    // if unable to eat wait to be signalled
    sem_wait(&boo[PhilNum]); /* Eat if enabled */

    sleep(1);
}

void drop_chopsticks(int PhilNum)
{
    sem_wait(&quantum); /* critical section */

    pflag[PhilNum] = THINKING;
    printf("....................................................................\n");
    printf(">>> Philosopher %s , id : %d, puting down Chopsticks %d and %d \n", P[PhilNum].name, PhilNum + 1, LEFT + 1, PhilNum + 1);
    printf("....................................................................\n");
    printf(">>> Philosopher %s , id : %d, is thinking. \n",P[PhilNum].name, PhilNum + 1);
    printf("....................................................................\n\n");

    test(LEFT); /* Let phil. on left eat if possible */
    test(RIGHT); /* Let phil. on rght eat if possible */
    sem_post(&quantum); /* up critical section */
}
```

```c

void* philospher(void* num)
{
    while (1) {
        int  i = (int)num;
        sleep(1);
        take_chopsticks(i) ;
        sleep(0);
        drop_chopsticks(i);
    }
}

int main()
{
    pthread_t Thread[5];

    // initialize the values to the semaphores

    //initially to 1, for mutual exclusion
    sem_init(&quantum, 0, 1);

    //semaphore boo[5] will be initially 0, for synchronization
    for (int i = 0; i < 5; i++){
        sem_init(&boo[i], 0, 0);
    }


    // creating philosopher processes

    for (int i = 0; i < 5; i++) {
        pthread_create(&Thread[i], NULL,philospher, (void*)P[i].id);
        printf("....................................................................\n");
        printf(">>> Philosopher %s , id : %d, is thinking. \n",P[i].name, i + 1);
        printf("....................................................................\n");
    }

    for (int i = 0; i < 5; i++){
        pthread_join(Thread[i], NULL);
    }
    return 0;
}
```

SCREENSHOT OF OUTPUT:



```
lamecodes@lamecodes-Inspiron-3521:~/Documents$ g++ main.c -lpthread
main.c:43:1: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
 };
 ^
main.c:43:1: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
main.c:43:1: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
main.c:43:1: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
main.c:43:1: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
lamecodes@lamecodes-Inspiron-3521:~/Documents$ ./a.out
....................................................................
>>> Philosopher Philosopher A , id : 1, is thinking.
....................................................................
....................................................................
>>> Philosopher Philosopher B , id : 2, is thinking.
....................................................................
....................................................................
>>> Philosopher Philosopher C , id : 3, is thinking.
....................................................................
....................................................................
>>> Philosopher Philosopher D , id : 4, is thinking.
....................................................................
....................................................................
>>> Philosopher Philosopher E , id : 5, is thinking.
....................................................................
....................................................................
>>> Philosopher Philosopher A , id : 1, is Hungry.
....................................................................

....................................................................
>>> Philosopher Philosopher C , id : 3, is Hungry.
....................................................................

....................................................................
>>> Philosopher Philosopher D , id : 4, is Hungry.
....................................................................

....................................................................
>>> Philosopher Philosopher E , id : 5, is Hungry.
....................................................................

....................................................................
>>> Philosopher Philosopher E ,id : 5,
 Picking up Chopsticks 4 and 5
....................................................................

....................................................................
```



```
>>> Philosopher Philosopher E ,id : 5,
 Picking up Chopsticks 4 and 5
....................................................................

....................................................................
>>> Philosopher Philosopher E , id : 5, is Eating.
....................................................................

....................................................................
>>> Philosopher Philosopher B , id : 2, is Hungry.
....................................................................

....................................................................
>>> Philosopher Philosopher B ,id : 2,
 Picking up Chopsticks 1 and 2
....................................................................

....................................................................
>>> Philosopher Philosopher B , id : 2, is Eating.
....................................................................

....................................................................
>>> Philosopher Philosopher E , id : 5, puting down Chopsticks 4 and 5
....................................................................
>>> Philosopher Philosopher E , id : 5, is thinking.
....................................................................

....................................................................
>>> Philosopher Philosopher D ,id : 4,
 Picking up Chopsticks 3 and 4
....................................................................

....................................................................
>>> Philosopher Philosopher D , id : 4, is Eating.
....................................................................

....................................................................
>>> Philosopher Philosopher B , id : 2, puting down Chopsticks 1 and 2
....................................................................
>>> Philosopher Philosopher B , id : 2, is thinking.
....................................................................

....................................................................
>>> Philosopher Philosopher A ,id : 1,
 Picking up Chopsticks 5 and 1
....................................................................
```

And this will continue running as there is no deadlock.

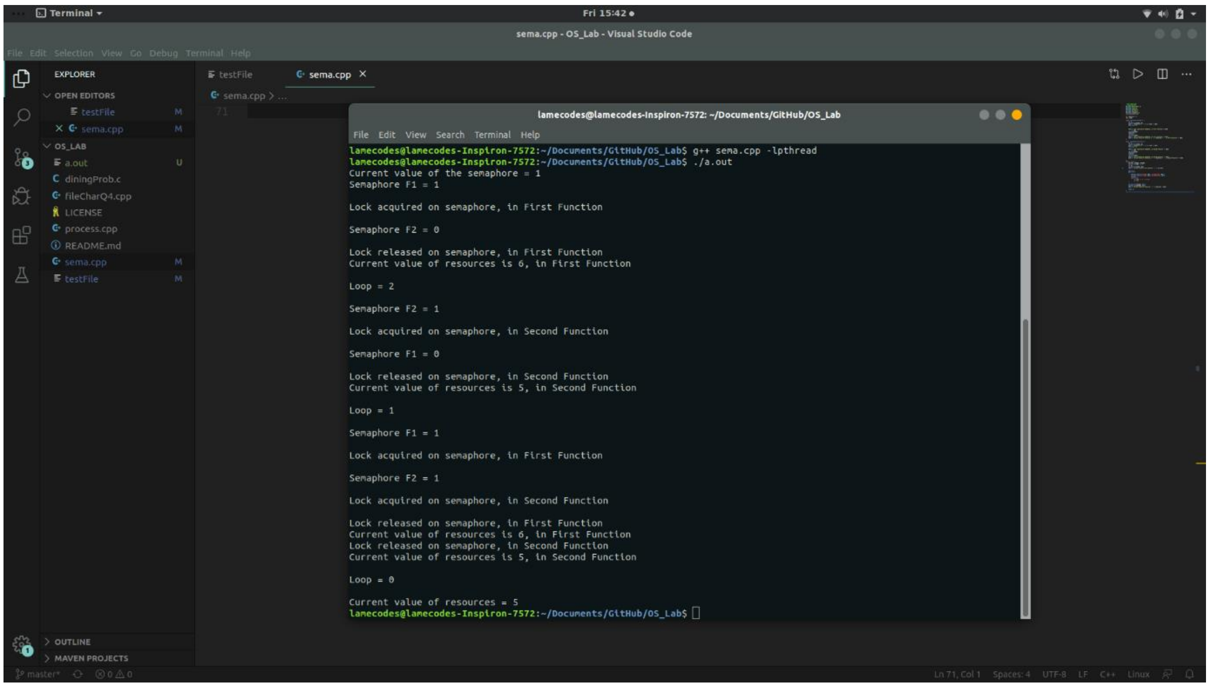
Question 8 b). Write a program to implement race condition using semaphores.

GITHUB LINK: https://github.com/Rudrakaniya/OS_Lab/blob/master/sema.cpp

CODE:

```cpp
#include <bits/stdc++.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>
using namespace std;

int resources = 5;
sem_t sema;

void * firstFunction(void *) {
    int q;
    sem_getvalue(&sema, &q);
    cout << "Semaphore F1 = " << q << endl << endl;
    while (q <= 0)
        ;

    cout << "Lock acquired on semaphore, in First Function" << endl
         << endl;
    sem_wait(&sema);
    resources++;
    sleep(2);
    sem_post(&sema);
    cout << "Lock released on semaphore, in First Function" << endl;
    cout << "Current value of resources is " << resources << ", in First Function" << endl;
}

void * secondFunction(void *) {
    int p;
    sem_getvalue(&sema, &p);
    cout << "Semaphore F2 = " << p << endl << endl;
    while (p <= 0)
        ;
    cout << "Lock acquired on semaphore, in Second Function" << endl
         << endl;
    sem_wait(&sema);
    resources--;
    sleep(2);
    sem_post(&sema);
    cout << "Lock released on semaphore, in Second Function" << endl;
    cout << "Current value of resources is " << resources << ", in Second Function" << endl;

}
int32_t main() {
    pthread_t thread1, thread2;
    sem_init( & sema, 0, 1);
    int sg;
    sem_getvalue(&sema, &sg);
    cout << "Current value of the semaphore = " << sg<<endl;

    int i = 3;
    while (i--)
    {
        pthread_create( & thread1, NULL, firstFunction, NULL);
        pthread_create( & thread2, NULL, secondFunction, NULL);
        sleep(5);
        cout << endl
             << "Loop = " << i << endl
             << endl;
    }

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    cout << "Current value of resources = " << resources << endl;

    return 0;
}
```

SCREENSHOT OF OUTPUT:



```
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ g++ sema.cpp -lpthread
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ ./a.out
Current value of the semaphore = 1
Semaphore F1 = 1

Lock acquired on semaphore, in First Function

Semaphore F2 = 0

Lock released on semaphore, in First Function
Current value of resources is 6, in First Function

Loop = 2

Semaphore F2 = 1

Lock acquired on semaphore, in Second Function

Semaphore F1 = 0

Lock released on semaphore, in Second Function
Current value of resources is 5, in Second Function

Loop = 1

Semaphore F1 = 1

Lock acquired on semaphore, in First Function

Semaphore F2 = 1

Lock acquired on semaphore, in Second Function

Lock released on semaphore, in First Function
Current value of resources is 6, in First Function
Lock released on semaphore, in Second Function
Current value of resources is 5, in Second Function

Loop = 0

Current value of resources = 5
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$
```

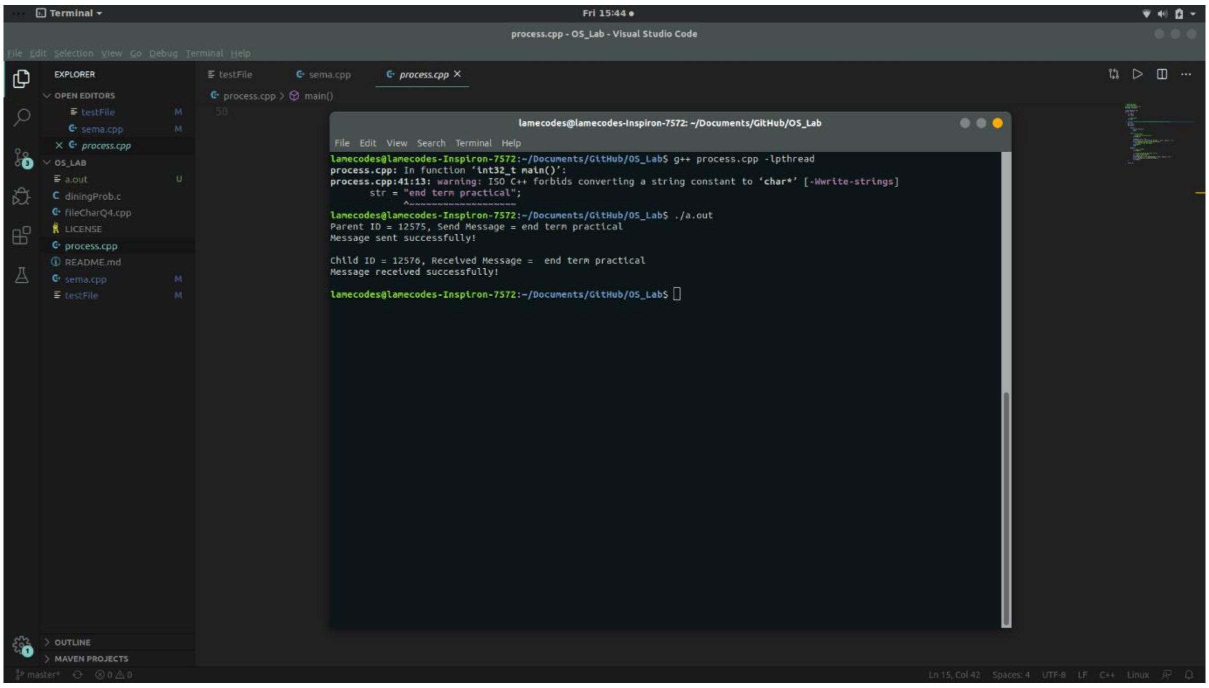
Question 8 c). Write a program to send a message "end term practical" from parent process to child process.

GITHUB LINK: https://github.com/Rudrakaniya/OS_Lab/blob/master/process.cpp

CODE:

```cpp
#include <bits/stdc++.h>
#include <unistd.h>

using namespace std;
int32_t main()
{
    int fd[2];
    char *str;

    // create pipe
    pipe(fd);

    // fork() returns 0 for child process, child-pid for parent process.
    pid_t p;
    p = fork();
    switch(p){

        case -1:
            printf("Error\n");
            break;

        case 0:
            // child process
            //closing the write descriptor
            close(fd[1]);

            read(fd[0], &str, 19);
            printf("Child ID = %d, Received Message =  %s\n", getpid(), str);
            printf("Message received successfully!\n\n");
            // closeing the read descriptor
            close(fd[0]);
            break;

        default:
             // parent process
            close(fd[0]);

            // send the string to the child process
            str = "end term practical";
            write(fd[1], &str, 19);
            printf("Parent ID = %d, Send Message = %s\n", getpid(), str);
            printf("Message sent successfully!\n\n");
            close(fd[1]);
    }

    return 0;
}
```

SCREENSHOT OF OUTPUT:



```
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ g++ process.cpp -lpthread
process.cpp: In function ‘int32_t main()’:
process.cpp:41:13: warning: ISO C++ forbids converting a string constant to ‘char*’ [-Wwrite-strings]
       str = "end term practical";
             ^~~~~~~~~~~~~~~~~~~~
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ ./a.out
Parent ID = 12575, Send Message = end term practical
Message sent successfully!

Child ID = 12576, Received Message =  end term practical
Message received successfully!

lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ 
```

Terminal
Fri 15:44
process.cpp - OS_Lab - Visual Studio Code
EXPLORER
OPEN EDITORS
testFile
sema.cpp
process.cpp
OS_LAB
a.out
diningProb.c
fileCharQ4.cpp
LICENSE
process.cpp
README.md
sema.cpp
testFile
OUTLINE
MAVEN PROJECTS
lamecodes@lamecodes-Inspiron-7572: ~/Documents/GitHub/OS_Lab
File Edit View Search Terminal Help
Ln 15, Col 42 Spaces: 4 UTF-8 LF C++ Linux

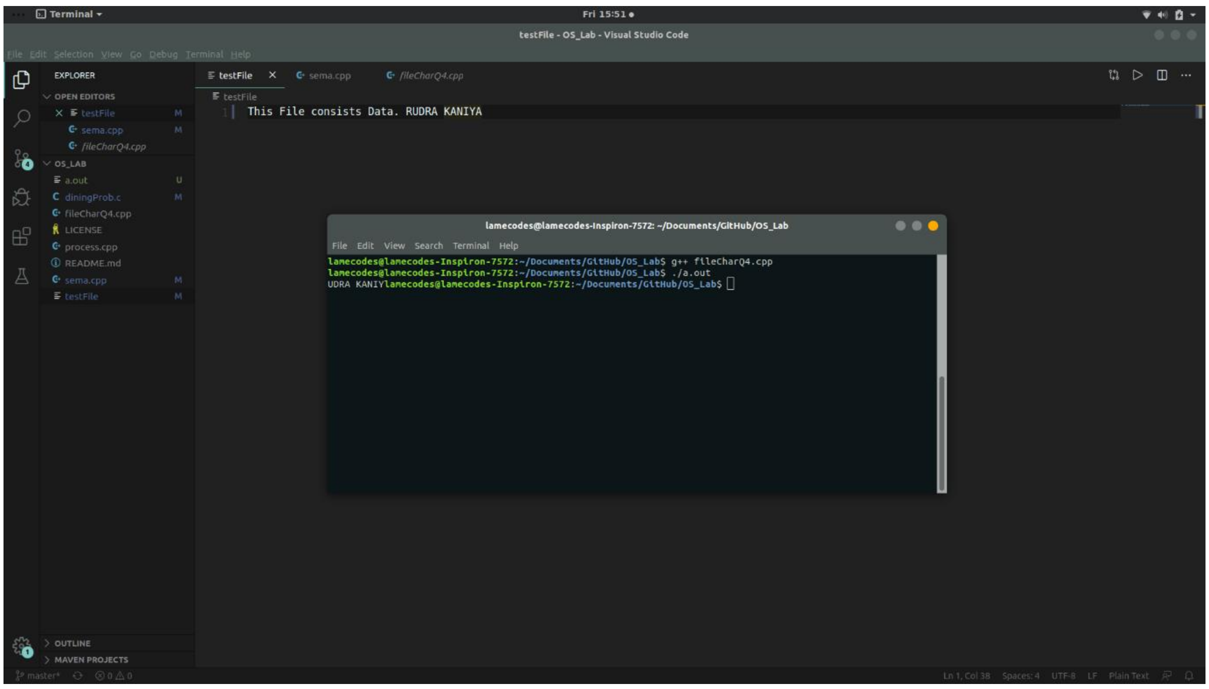
Question 8 d). Write a program to display the last 10 characters of file on screen.

GITHUB LINK: https://github.com/Rudrakaniya/OS_Lab/blob/master/fileCharQ4.cpp

CODE:

```c
#include<stdio.h>
#include<unistd.h>
#include<fcntl.h>
#include<errno.h>
int main()
{
    int len, fd;
    char str[30];
    fd = open("testFile", O_RDONLY, 0777);
    if (fd == -1)
        perror("Error:");

    lseek(fd, -11, SEEK_END);
    len = read(fd, str, 10);
    write(1, str, len);

    return 0;
}
```

SCREENSHOT OF OUTPUT:



```
This File consists Data. RUDRA KANIYA
```

```
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ g++ fileCharQ4.cpp
lamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ ./a.out
UDRA KANIYlamecodes@lamecodes-Inspiron-7572:~/Documents/GitHub/OS_Lab$ 
```