OPERATING SYSTEM LABORATORY WORKSHEET

NAME: Shivam Kumar Singh	**REGISTRATION NUMBER:** 11803277
ROLLNO: B-61	**QUESTION NO:** 1

Format of the report:
Text Size: 10
Text Style: Times New Roman

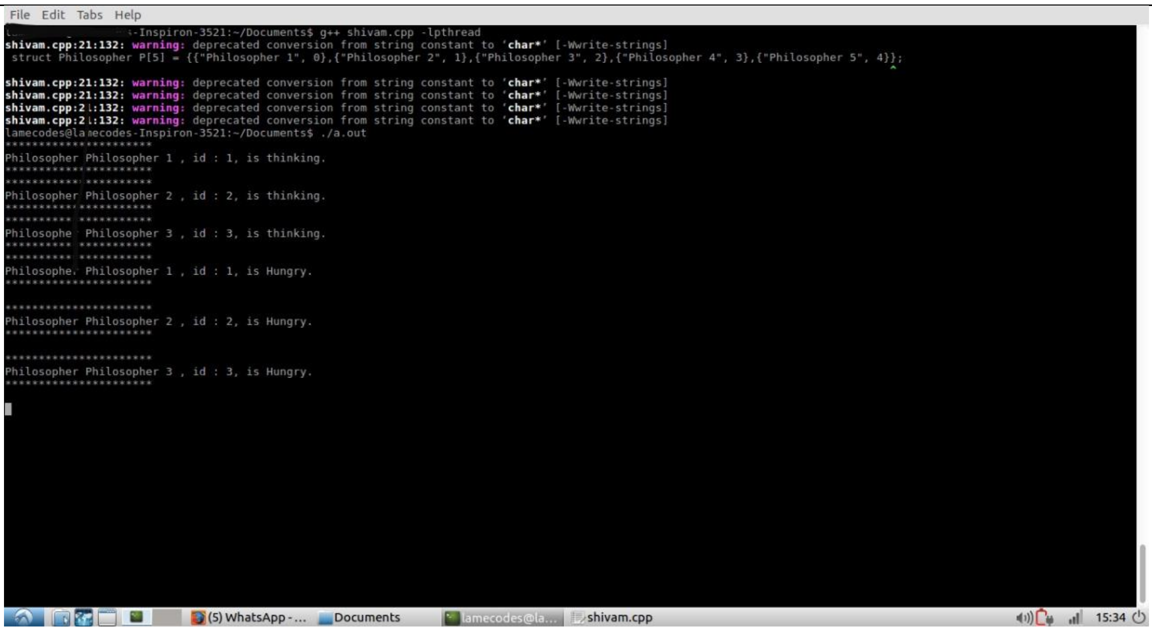
PART	GITHUB LINK,CODE,SCREENSHOT OF OUTPUT
a	**GITHUB LINK:** https://github.com/shivam9608/OSlaboratory

CODE:

```
/* Authored by: Shivam Kumar Singh (Roll: B-61)
            Section: K18MS (Reg. No: 11803277)

             (a) DINING PHILOSOPHER PROBLEM (Using Semaphore)                */

#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]={
    {
        "Professor A",0
    },
    {
        "Professor B",1
    },
    {
        "Professor C",2
    },
    {
       "Professor D",3
    },
    {
       "Professor E",4
    }
};

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


void test(int PhilNum)
{
   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 */
}

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:



```
[redacted]-Inspiron-3521:~/Documents$ g++ shivam.cpp -lpthread
shivam.cpp:21:132: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
 struct Philosopher P[5] = {{"Philosopher 1", 0},{"Philosopher 2", 1},{"Philosopher 3", 2},{"Philosopher 4", 3},{"Philosopher 5", 4}};
                                                                                                                                   ^
shivam.cpp:21:132: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
shivam.cpp:21:132: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
shivam.cpp:21:132: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
shivam.cpp:21:132: warning: deprecated conversion from string constant to 'char*' [-Wwrite-strings]
lamecodes@lamecodes-Inspiron-3521:~/Documents$ ./a.out
**********************
Philosopher Philosopher 1 , id : 1, is thinking.
**********************
**********************
Philosopher Philosopher 2 , id : 2, is thinking.
**********************
**********************
Philosopher Philosopher 3 , id : 3, is thinking.
**********************
**********************
Philosopher Philosopher 1 , id : 1, is Hungry.
**********************

**********************
Philosopher Philosopher 2 , id : 2, is Hungry.
**********************

**********************
Philosopher Philosopher 3 , id : 3, is Hungry.
**********************

```

b

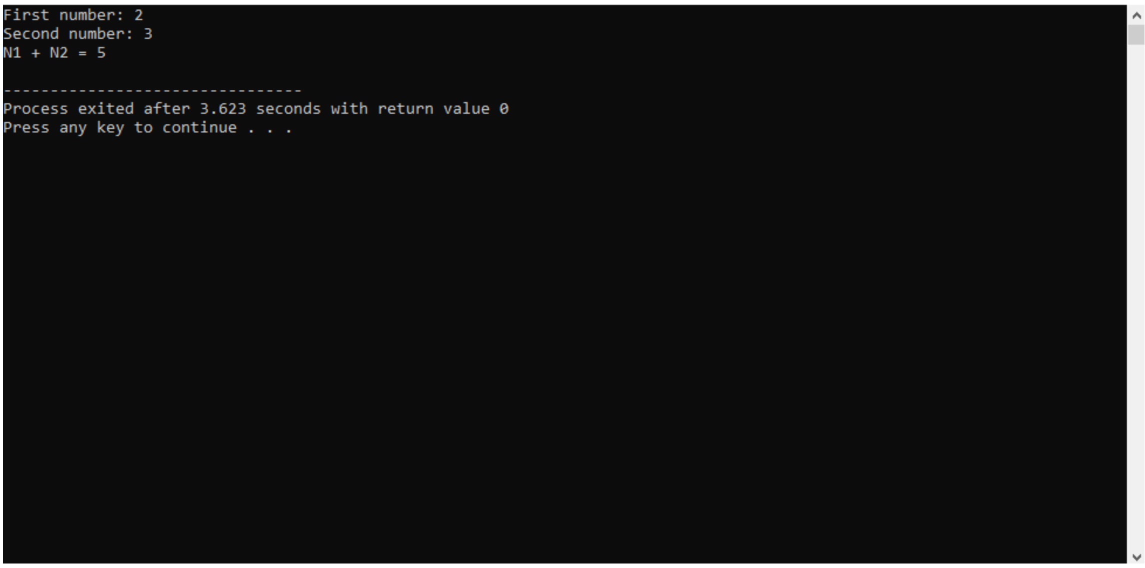
GITHUB LINK: https://github.com/shivam9608/OSlaboratory

CODE:

```
/* Authored by: Shivam Kumar Singh (Roll: B-61)
            Reg. No. 11803277 (Section: K18MS)

(b) ADDITION OF TWO NUMBERS USING THREADS */

#include<stdio.h>
#include<pthread.h>
int global[2];

void *sum_thread(void *arg)
{
    int *args_array;
    args_array = arg;

    int n1,n2,sum;
    n1=args_array[0];
    n2=args_array[1];
    sum = n1+n2;

    printf("N1 + N2 = %d\n",sum);
}

int main()
{
    printf("First number: ");
    scanf("%d",&global[0]);


    printf("Second number: ");
    scanf("%d",&global[1]);
    pthread_t tid_sum;
    pthread_create(&tid_sum,NULL,sum_thread,global);
    pthread_join(tid_sum,NULL);
```

```
   return 0;
}
```

SCREENSHOT OF OUTPUT:



```
First number: 2
Second number: 3
N1 + N2 = 5

--------------------------------
Process exited after 3.623 seconds with return value 0
Press any key to continue . . .
```

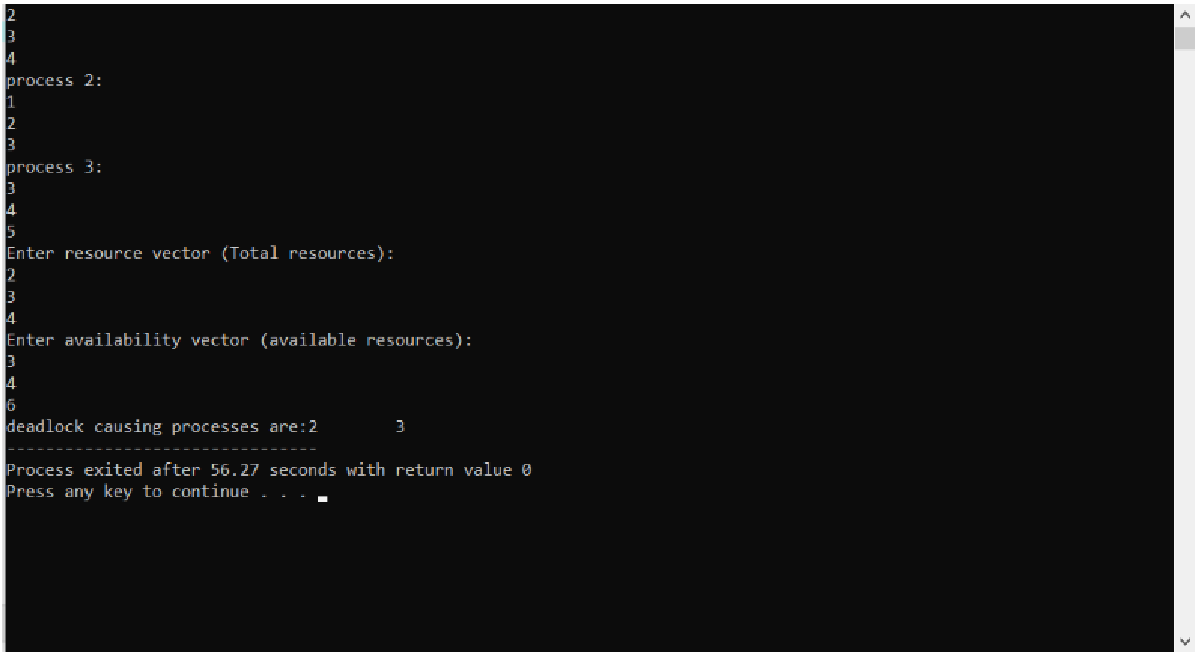
C

GITHUB LINK: https://github.com/shivam9608/OSlaboratory

CODE:

```
/* Authored by: Shivam Kumar Singh (Roll: B-61)
         Section: K18MS (Reg. No: 11803277)

          (d) TO SIMULATE DEADLOCK SITUATION              */

#include<iostream>
#include<thread>
#include<mutex>
using namespace std;
std::mutex m1;
std::mutex m2;
std::mutex m3;
void thread1() {
          m1.lock();
          m2.lock();
   m3.lock();
          cout<<"Critical section of Thread Thread One\n";
          m1.unlock();
          m2.unlock();
   m3.unlock();
}
void thread2() {
          m2.lock();
          m1.lock();
   m3.lock();
          cout<<"Critical section of Thread Thread Two\n";
          m2.unlock();
          m1.unlock();
   m3.unlock();
}
void thread3() {
   m3.lock();
   m1.lock();
   m2.lock();
   cout<<"Critical section of Thread Thread Three\n";
   m3.unlock();
   m1.unlock();
   m2.unlock();
}
int main()
{
           thread t1(thread1);
           thread t2(thread2);
           thread t3(thread3);
           t1.join();
           t2.join();
           t3.join();
           return 0;
}
```

SCREENSHOT OF OUTPUT:



```
2
3
4
process 2:
1
2
3
process 3:
3
4
5
Enter resource vector (Total resources):
2
3
4
Enter availability vector (available resources):
3
4
6
deadlock causing processes are:2        3
--------------------------------
Process exited after 56.27 seconds with return value 0
Press any key to continue . . .
```

d

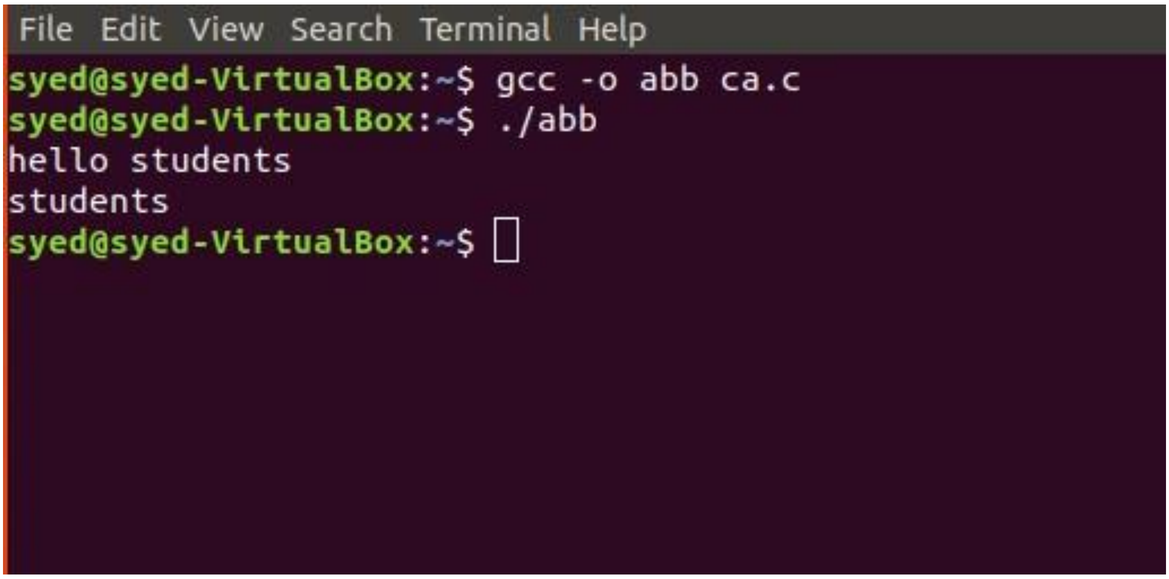
GITHUB LINK: https://github.com/shivam9608/OSlaboratory

CODE:

```
/* Authored by: Shivam Kumar Singh (Roll: B-61)
          Section: K18MS (Reg. No: 11803277)

           (d) TO COPY CONTENT OF ONE FILE INTO OTHER                  */

           #include <unistd.h>
           #include<stdio.h>
           #include<fcntl.h>
```

```
int main()
{
        int fd, n, p;
        char arr[100];
        fd = open("SEEK_END.txt", O_CREAT|O_RDWR, 0777);
        n = read(0, arr, 100);
        write(fd, arr, n);
        p = lseek(fd, -5, SEEK_END);
        read(fd, arr, 5);
        write(1, arr, 5);
        printf("\n");
}
```

SCREENSHOT OF OUTPUT:



```
syed@syed-VirtualBox:~$ gcc -o abb ca.c
syed@syed-VirtualBox:~$ ./abb
hello students
students
syed@syed-VirtualBox:~$ 
```