OPERATING SYSTEM LABORATORY WORKSHEET

NAME: Anamika Biswas	REGISTRATION NUMBER: 11807415
ROLLNO:K18NZA02	QUESTION NO:2

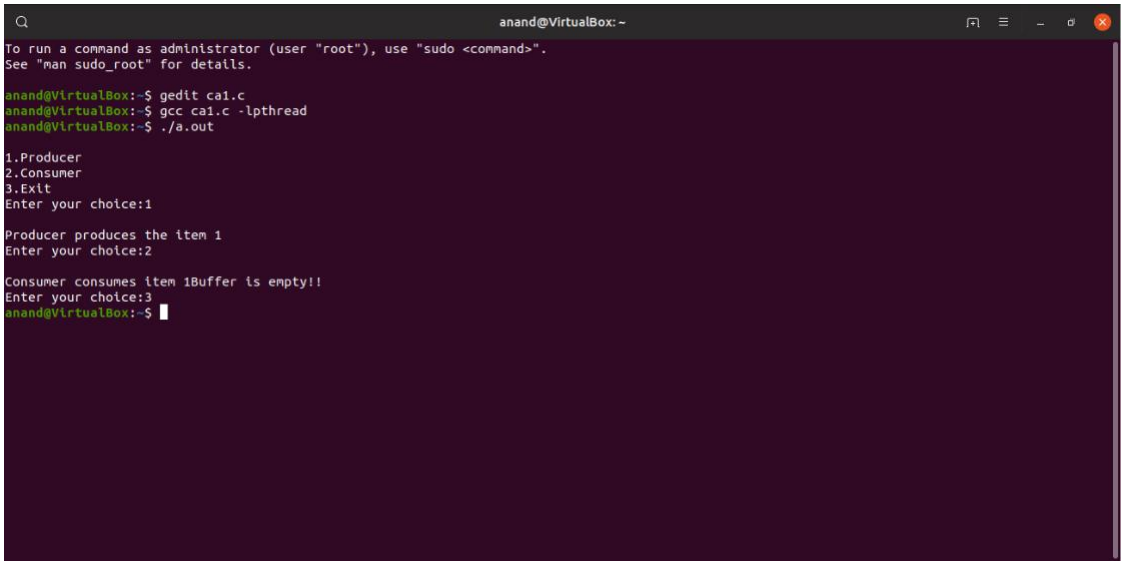
GITHUBLINK:
https://github.com/anamika2511/operating-system

PART	GITHUB LINK,CODE,SCREENSHOT OF OUTPUT

a	a) Write a program to implement the solution of Producer Consumer problem using semaphores.
	
	CODE:
	#include<stdio.h>
	#include<stdlib.h>
	int mutex=1,full=0,empty=3,x=0;
	int main()
	{
	int n;
	void producer();
	void consumer();
	int wait(int);
	int signal(int);
	printf("\n1.Producer\n2.Consumer\n3.Exit");
	while(1)
	{
	printf("\nEnter your choice:"); scanf("%d",&n);
	switch(n)
	{
	case 1:
	if((mutex==1)&&(empty!=0))
	producer();
	else
	printf("Buffer is full!!");
	break;
	case 2:
	if((mutex==1)&&(full!=0))
	consumer();
	else;
	printf("Buffer is empty!!");
	break;
	case 3:
	exit(0);
	break;
	}
	}
	return 0;

```
}
int wait(int s)
{
return (--s);
}
int signal(int s)
{
return(++s);
}
void producer()
{
mutex=wait(mutex);
full=signal(full);
empty=wait(empty);
x++;
printf("\nProducer produces the item %d",x);
mutex=signal(mutex);
}
void consumer()
{
mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\nConsumer consumes item %d",x);
x--;
mutex=signal(mutex);
}
```

SCREENSHOT OF OUTPUT:



b

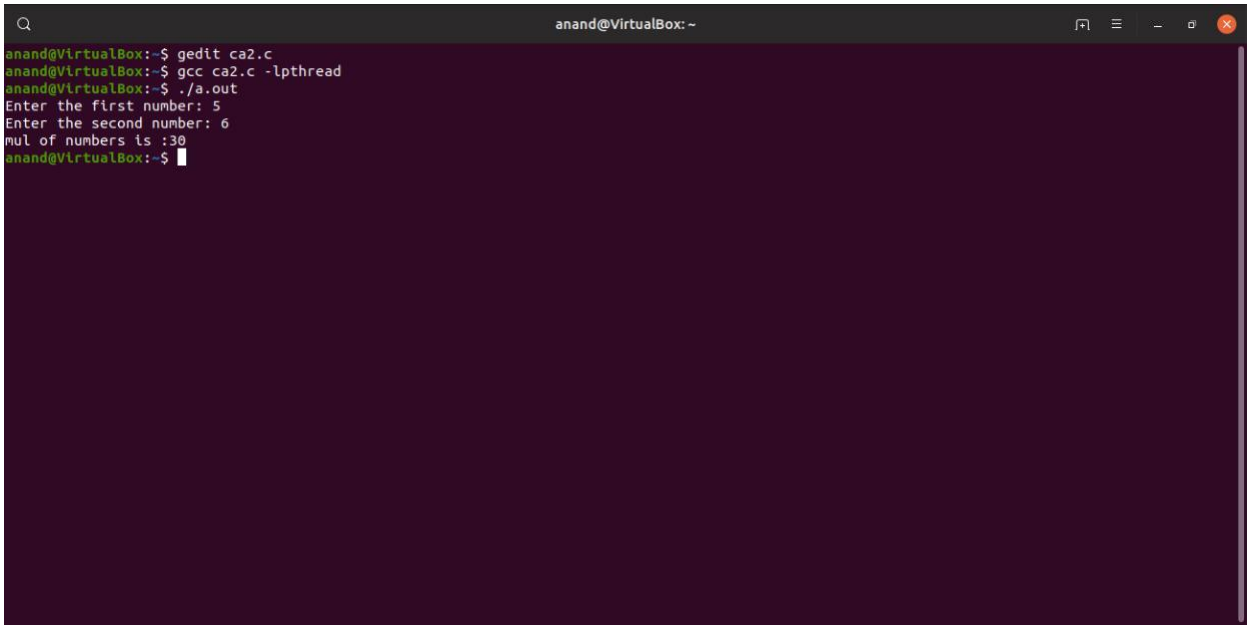
CODE(B)wap to display multiplication of numbers using threads

```
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
void* fun(void* arg)
{
```

```
int *a=(int *)arg;
int mul=1;
for(int i=0;i<2;i++)
{
mul=mul*a[i];
}
printf("mul of numbers is :%d\n",mul);
}

int main()
{
pthread_t p;
int n1,n2;
printf("Enter the first number: ");
scanf("%d",&n1);
printf("Enter the second number: ");
scanf("%d",&n2);
int a[]={n1,n2};
pthread_create(&p,NULL,fun,(void*)a);
pthread_join(p,NULL);
}
```

SCREENSHOT OF OUTPUT



:

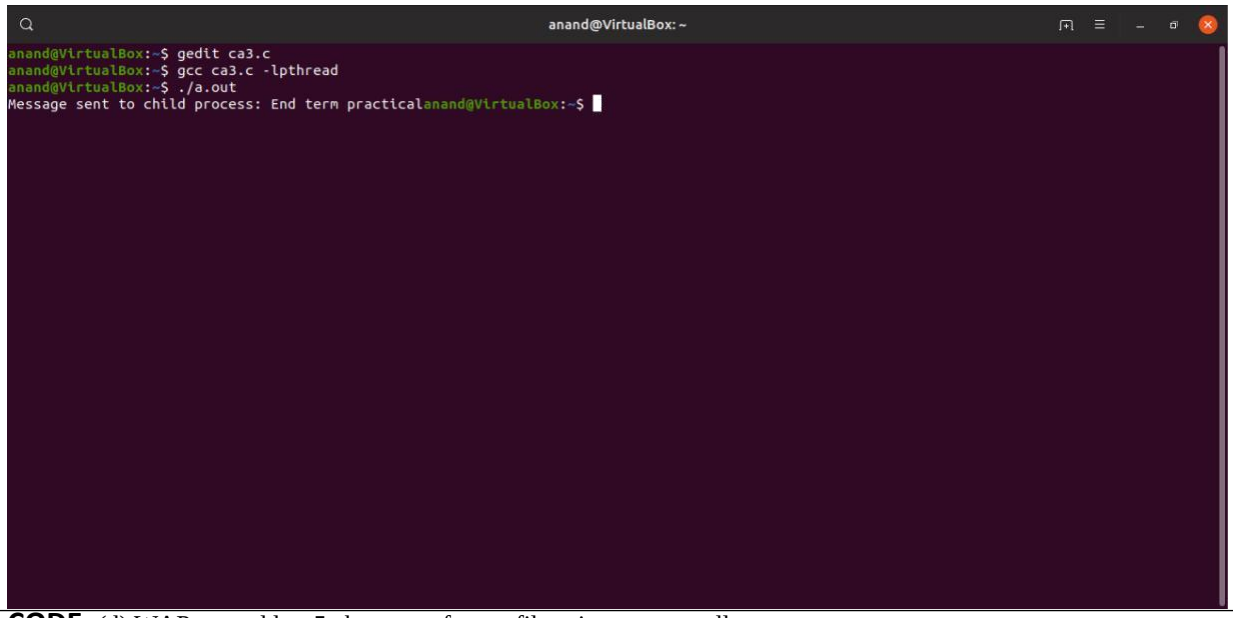
C.

CODE: (c) write a program to send a message "end term practical" from parent process to child process.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<string.h>
#include<sys/wait.h>
int main(void)
```

```
{
pid_t pid;
int r;
char buff[1024];
char *cp;
int readpipe[2];
int a;
a=pipe(readpipe);
pid = fork();
if(pid == 0)
{
wait(0);
read(readpipe[0],buff,sizeof(buff));
printf("Message sent to child process: %s",buff);
}
else
{
cp="End term practical";
write(readpipe[1],cp,strlen(cp)+1);
wait(0);
}
return 0;
}
```

SCREENSHOT OF OUTPUT:



D. **CODE:** (d) WAP to read last 5 characters from a file using system calls

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

int main()
{
int fd,n;
char buff[100];
fd=open("anand",O_CREAT|O_RDWR|O_APPEND,0777);
```

```
n=read(0,buff,100);
write(fd,buff,n);
lseek(fd,-6,SEEK_END);
read(fd,buff,5);
write(1,buff,5);
printf("\n");
}
```

SCREENSHOT OF OUTPUT: