

**PDEU** PANDIT  
DEENDAYAL  
ENERGY  
UNIVERSITY



**NAME : MEET SHELDIYA**

**ROLL NUMBER : 19BIT076**

**SUBJECT : OPERATING SYSTEM LAB (20IC301P)**

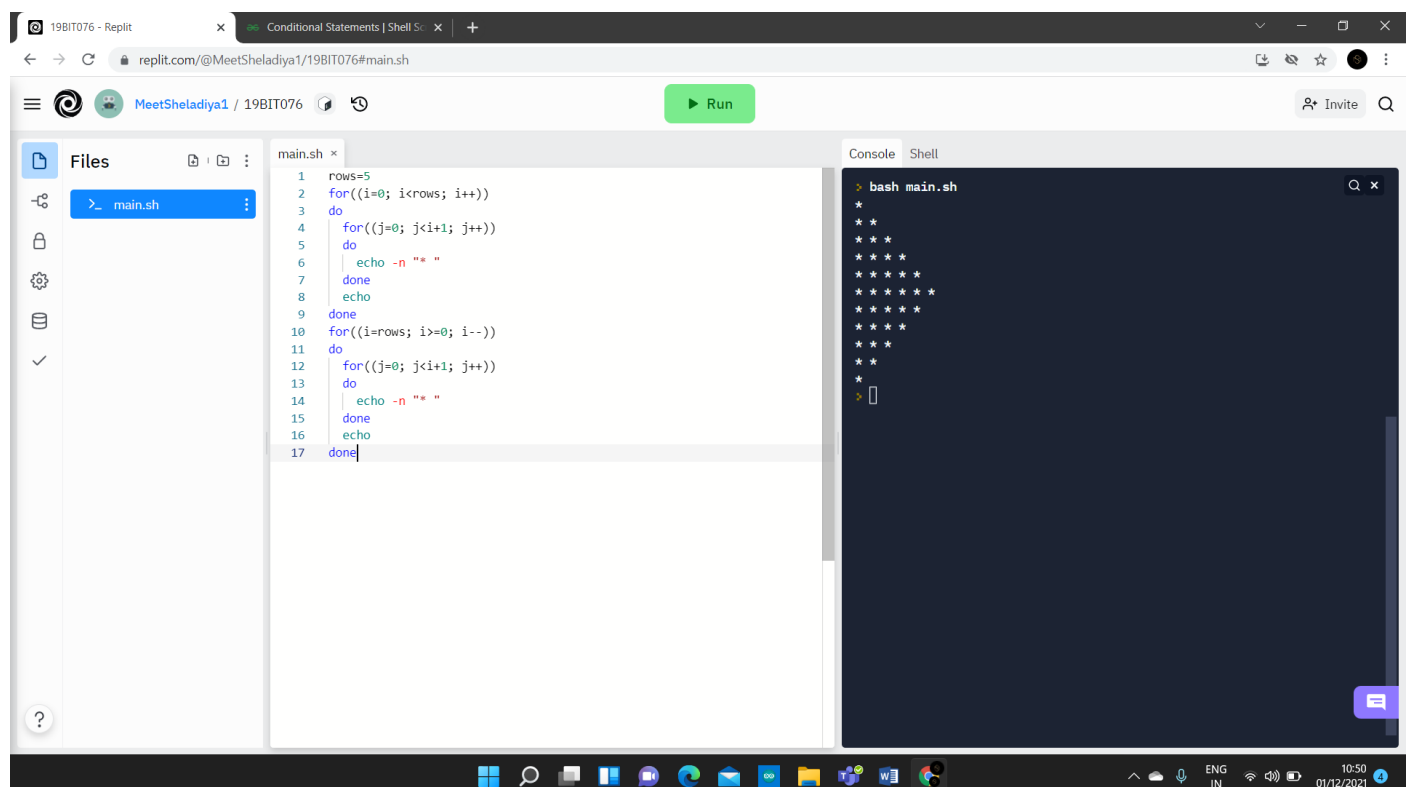
**BRANCH : INFORMATION AND COMMUNICATION  
TECHNOLOGY (I.C.T.) DIVISION-2**

**1.a)** Write a shell script to print the following pattern

### Code:

```
rows=5
for((i=0; i<rows; i++))
do
    for((j=0; j<i+1; j++))
    do
        echo -n "* "
    done
    echo
done
for((i=rows; i>=0; i--))
do
    for((j=0; j<i+1; j++))
    do
        echo -n "* "
    done
    echo
done
```

### Output:



The screenshot shows a Replit environment with a file named `main.sh` and a console window. The script in `main.sh` is as follows:

```
1 rows=5
2 for((i=0; i<rows; i++))
3 do
4     for((j=0; j<i+1; j++))
5     do
6         echo -n "* "
7     done
8     echo
9 done
10 for((i=rows; i>=0; i--))
11 do
12     for((j=0; j<i+1; j++))
13     do
14         echo -n "* "
15     done
16     echo
17 done
```

The console window shows the output of the script, which is a diamond-shaped pattern of asterisks:

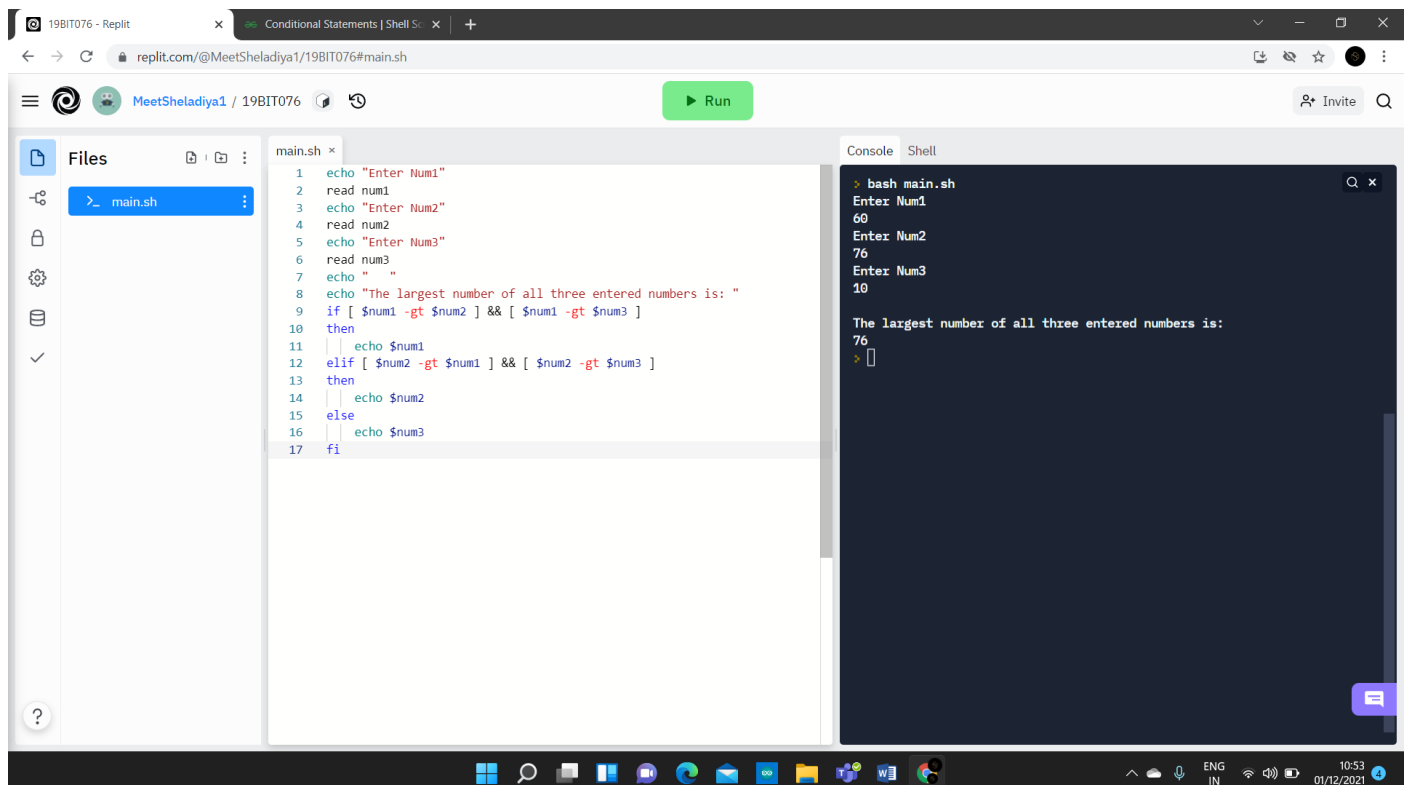
```
> bash main.sh
*
* *
* * *
* * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
>
```

**1.b)** Write a shell script to find the largest of three numbers.

### Code:

```
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3
echo " "
echo "The largest number of all three entered numbers is: "
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo $num2
else
    echo $num3
fi
```

### Output:



The screenshot shows a Replit environment with a file named `main.sh` containing the following shell script:

```
1 echo "Enter Num1"
2 read num1
3 echo "Enter Num2"
4 read num2
5 echo "Enter Num3"
6 read num3
7 echo " "
8 echo "The largest number of all three entered numbers is: "
9 if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
10 then
11     echo $num1
12 elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
13 then
14     echo $num2
15 else
16     echo $num3
17 fi
```

The console output shows the script being executed with the following input and output:

```
> bash main.sh
Enter Num1
60
Enter Num2
76
Enter Num3
10

The largest number of all three entered numbers is:
76
>
```

## 2. Implement the Dining philosophers problem using semaphore.

### Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#include<semaphore.h>
#include<unistd.h>
sem_t chopstick[5];
void * philos(void *);
void eat(int);
int main()
{
    int i,n[5];
    pthread_t T[5];
    for(i=0;i<5;i++)
        sem_init(&chopstick[i],0,1);
    for(i=0;i<5;i++){
        n[i]=i;
        pthread_create(&T[i],NULL,philos,(void *)&n[i]);
    }
    for(i=0;i<5;i++)
        pthread_join(T[i],NULL);
}
void * philos(void * n)
{
    int ph=*(int *)n;
    printf("Philosopher %d wants to eat\n",ph);
    printf("Philosopher %d tries to pick left chopstick\n",ph);
    sem_wait(&chopstick[ph]);
    printf("Philosopher %d picks the left chopstick\n",ph);
    printf("Philosopher %d tries to pick the right chopstick\n",ph);
    sem_wait(&chopstick[(ph+1)%5]);
    printf("Philosopher %d picks the right chopstick\n",ph);
    eat(ph);
    sleep(2);
    printf("Philosopher %d has finished eating\n",ph);
```

```

sem_post(&chopstick[(ph+1)%5]);

printf("Philosopher %d leaves the right chopstick\n",ph);

sem_post(&chopstick[ph]);

printf("Philosopher %d leaves the left chopstick\n",ph);

}

void eat(int ph)
{
    printf("Philosopher %d begins to eat\n",ph);
}

```

## Output:

```

C:\Users\Lenovo\Desktop\Meet\Sem_OS\OS\19BIT076.exe
Philosopher 0 wants to eat
Philosopher 2 wants to eat
Philosopher 2 tries to pick left chopstick
Philosopher 2 picks the left chopstick
Philosopher 2 tries to pick the right chopstick
Philosopher 2 picks the right chopstick
Philosopher 2 begins to eat
Philosopher 0 tries to pick left chopstick
Philosopher 0 picks the left chopstick
Philosopher 0 tries to pick the right chopstick
Philosopher 0 picks the right chopstick
Philosopher 0 begins to eat
Philosopher 4 wants to eat
Philosopher 4 tries to pick left chopstick
Philosopher 4 picks the left chopstick
Philosopher 4 tries to pick the right chopstick
Philosopher 1 wants to eat
Philosopher 1 tries to pick left chopstick
Philosopher 3 wants to eat
Philosopher 3 tries to pick left chopstick
Philosopher 2 has finished eating
Philosopher 0 has finished eating
Philosopher 0 leaves the right chopstick
Philosopher 0 leaves the left chopstick
Philosopher 1 picks the left chopstick
Philosopher 1 tries to pick the right chopstick
Philosopher 4 picks the right chopstick
Philosopher 4 begins to eat
Philosopher 2 leaves the right chopstick
Philosopher 2 leaves the left chopstick
Philosopher 3 picks the left chopstick
Philosopher 3 tries to pick the right chopstick
Philosopher 1 picks the right chopstick
Philosopher 1 begins to eat
Philosopher 1 has finished eating
Philosopher 1 leaves the right chopstick
Philosopher 1 leaves the left chopstick
Philosopher 4 has finished eating
Philosopher 4 leaves the right chopstick
Philosopher 4 leaves the left chopstick
Philosopher 3 picks the right chopstick
Philosopher 3 begins to eat
Philosopher 3 has finished eating
Philosopher 3 leaves the right chopstick
Philosopher 3 leaves the left chopstick

Process returned 0 (0x0)   execution time : 6.066 s
Press any key to continue.

```