ASSIGNMENT : CSE316

INTEGRATED B.TECH.-M.TECH.

in

COMPUTER SCIENCE AND ENGINEERING

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Submitted By

Akhil Pathania

11804258

Roll No: 33

Section: K18ZV

Submitted to

Dr. Priyanka Mittal

1) Q 20: There are 3 student processes and 1 teacher process. Students are supposed to do their assignments and they need 3 things for that pen, paper and question paper. The teacher has an infinite supply of all the three things. One students has pen, another has paper and another has question paper. The teacher places two things on a shared table and the student having the third complementary thing makes the assignment and tells the teacher on completion. The teacher then places another two things out of the three and again the student having the third thing makes the assignment and tells the teacher on completion. This cycle continues. WAP to synchronize the teacher and the students.

Code Snippet:-

Constraints:-

Here, NO defines the values cannot be less than 1. It means number of things should always be greater than or equal to 1. And we are given not name instance of that directly provided 1, 2, 3 numbers.

Boundary Conditions:-

Boundary condition for No of things is 1 and 2.

Code:-

```
#include<stdio.h>
#include<stdbool.h>
struct requirement
{
        bool pen;
        bool paper ;
        bool question_paper ;
        bool all_three ;
};
int main()
{
        int n=3;
```

```
struct requirement  s[n];
s[0].pen=true;
s[0].paper = false;
s[0].question_paper = false;
s[0].all_three= false;
s[1].pen=false;
s[1].paper = true;
s[1].question_paper = false;
s[1].all_three = false;
s[2].pen=false;
s[2].paper = false;
s[2].question_paper = true;
s[2].all_three = false   ;
while(s[0].all_three==false||s[1].all_three==false||s[2].all_three==false)
{
        int ch1,ch2;
        printf("\nResources:\n1.pen\n2.paper\n3.question paper\n Enter the two things which is to be placed on the shared table: ");
        scanf("%d%d",&ch1,&ch2);
        if(ch1==1 && ch2==2 && s[2].all_three==false)
        {
                s[2].all_three=true ;
                printf("Third Student has completed the task\n");
        }
        if(ch1==2 && ch2==3 && s[0].all_three==false)
        {
                s[0].all_three=true;
                printf("First Student has completed the task\n");
        }
```

```
            if(ch1==1 && ch2==3 && s[1].all_three==false)
            {
                s[1].all_three=true;
                printf("Second Student has completed the task\n");
            }
        }
        printf("All the students now have completed their respective tasks succesfully\n");
        return 0;
}
```

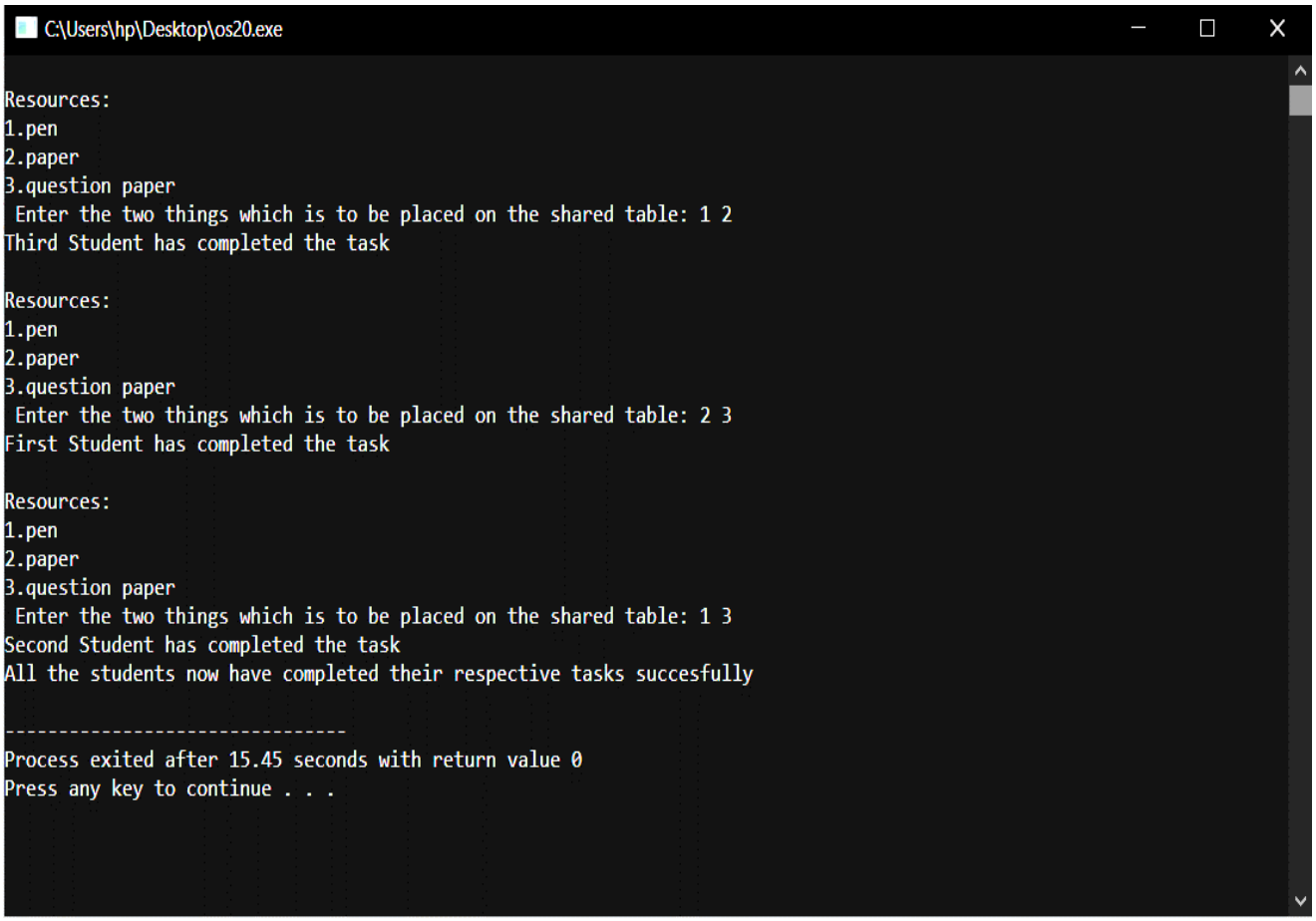
Test Cases:-

To check whether it works for a given input.

Input=1, 2 or 2, 3 or 1, 3

Process=It will process the above input and gives the output.

Output= Successful



Question : Researchers designed one system that classified interactive and noninteractive processes automatically by looking at the amount of terminal I/O. If a process did not input or output to the terminal in a 1-second interval, the process was classified as noninteractive and was moved to a lower-priority queue. In response to this policy, one programmer modified his programs to write an arbitrary character to the terminal at regular intervals of less than 1 second. The system gave his programs a high priority, even though the terminal output was completely meaningless.

Code Snippet:-

Constraints:-

Process must input or output to the terminal in a 1-second interval.

Boundary Conditions:-

The system gave his programs a high priority, even though the terminal output was completely meaningless.

CODE **:**

```
#include<stdio.h>
int main()
{
	int i, type[20],n;
	int resptime[20];
	printf("Number of process: ");
	scanf("%d",&n);
	printf("Enter the data\n");
	for(i=0;i<n;i++)
	{
		printf("Response time of P%d (in milliseconds): ",i);
		scanf("%d",&resptime[i]);
		if(resptime[i]<1000)
		{
			type[i]=1;
		}
		else
		{
			type[i]=0;
		}
	}
	printf("Process Number\tResponse Time\tType\t\tPriority");
	for(i=0;i<n;i++)
```

```
        {
                printf("\nP%d\t\t%dms\t\t",i,resptime[i]);
                if(type[i]==1)
                {
                        printf("Interactive\tHigh");
                }
                else
                {
                        printf("Non-Interactive\tLow");
                }
        }
}
```

TEST CASE :