**LOVELY PROFESSIONAL UNIVERSITY**

**Academic Task-3 (Operating System)**

School of Computer Science and Engineering ,Faculty of Technology And Sciences Name of the faculty member: M. Vijaya Raju

Course Code: CSE 316 Course Title: Operating System

Date of Submission: 03/04/2020

# Mention the below in header of the word document Student Name: Vaibhav Yelam

**Student ID: 11806530**

**Email Address:** [**vaibhavyelam02@gmail.com**](mailto:vaibhavyelam02@gmail.com)

**GitHub Link:** [**https://github.com/Vaibhav210/os-project.git**](https://github.com/Vaibhav210/os-project.git)

**project Code:**

7. Researchers designed one system that classified interactive and non-interactive 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 non-interactive 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.

#include<stdio.> int main()

{

int i, type[20],n; int resp\_time[20];

printf("enter the no,of processes you want: "); scanf("%d",&n);

printf("Enter any data\n"); scanf("%d",&n); for(i=0;i<n;i++)

{

printf("Response time of Process is %d (m\_sec) : ",i) scanf("%d",&resp\_time[i]);

if(resp\_time[i]<1000)

{

}

else

{

}

}

type[i]=1;

type[i]=0;

printf("\nProcess Number\t Response Time\t ProcessType\t Priority"); for(i=0;i<n;i++)

{

printf("\n%d\t\t%dms\t",i,resp\_time[i]); if(type[i]==1)

{

}

else

{

}

}

}

printf("Interactive\tHigh proiority");

printf("Non-Interactive\tLow priority--");

1. 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,an other 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.

#include<stdio.h> #include<stdbool.h>

struct apparatus

{

bool pen ; bool paper ;

bool question\_paper ; bool all\_three\_students ;

};

int main()

{

int n=3;

struct apparatus a[n]; a[0].pen=true; a[0].paper = false;

a[0].question\_paper = false; a[0].all\_three\_students= false; a[1].pen=false;

a[1].paper = true; a[1].question\_paper = false; a[1].all\_three\_students = false; a[2].pen=false;

a[2].paper = false; a[2].question\_paper = true; a[2].all\_three\_students = false ;

while(a[0].all\_three\_students==false||a[1].all\_three\_students==false||a[2].all\_three

\_students==false)

{

int b1,b2;

printf("\n Assigned things:\n1.pen\n2.paper\n3.question paper\n

Enter the two things that are placed on the shared table: "); scanf("%d%d",&b1,&b2);

if(b1==1 && b2==2 && a[2].all\_three\_students==false)

{

a[2].all\_three\_students=true ;

printf("1st Student completed the task\n");

}

if(b1==2 && b2==3 && a[0].all\_three\_students==false)

{

a[0].all\_three\_students=true;

printf("2nd Student completed the task\n");

}

if(b1==1 && b2==3 && a[1].all\_three\_students==false)

{

a[1].all\_three\_students=true;

printf("3rd Student completed the task\n");

# Explain the problem in terms of operating system concept? (Max 200 word)

**Description:**

**Q.No:7 :-**In this question I used th e concept of process scheduling.

* First it will the response time , if response time is equal to “0” then it will print **non interactive and low priority**. If response time is equal to “1” then it will print **interactive and high priority**.
* **Process scheduling:** The **process scheduling** is the activity of

the **process** manager that handles the removal of the running **process** from the CPU and the selection of another **process** on the basis of a particular strategy

* **Types of process scheduling:-** 1) First Come First Serve (FCFS), 2) Shortest- Job-First (SJF) **Scheduling** 3) Shortest Remaining Time 4) Priority **Scheduling** 5) Round Robin **Scheduling** 6) Multilevel Queue **Scheduling**.

# Write the algorithm for proposed solution of the assigned problem. Algorithm:

Step1: start

**Step2:** int i , type[20] ,n; Step3: for(i=0;i<n;i++)

Step4: if(resp\_time[i]<1000)

{

}

else

{

}

type[i]=1;

type[i]=0;

Step5 : for(i=0;i<n;i++) if(type[i]==1)

{

}

else

{

}

printf("Interactive\tHigh proiority");

printf("Non-Interactive\tLow priority--");

Step6: repeat step 3,step4,step5 Step7: exit

# Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity)

**Complexity for step 3 is n Complesity for step 5 is n Over all complexity is n2**

* **Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint.**

**Q.no :- 20**

* if(b1==1 && b2==2 && a[2].all\_three\_students==false)

{

a[2].all\_three\_students=true ;

printf("1st Student completed the task\n");

}

if(b1==2 && b2==3 && a[0].all\_three\_students==false)

{

a[0].all\_three\_students=true;

printf("2nd Student completed the task\n");

}

if(b1==1 && b2==3 && a[1].all\_three\_students==false)

{

a[1].all\_three\_students=true;

printf("3rd Student completed the task\n");

# Q.no:- 7

if(resp\_time[i]<1000)

{

}

else

{

}

if(type[i]==1)

{

}

else

{

}

type[i]=1;

type[i]=0;

printf("Interactive\tHigh proiority");

printf("Non-Interactive\tLow priority--");

# Explain the boundary conditions of the implemented code. Description:

**Q.no:- 7**

resp\_time[i]<1000

**If** response time is less than 1000 it will assign interactive and high priority , if response time is greater than 1000 it will assign non interactive and low priority.

# .Explain all the test cases applied on the solution of assigned problem.

**Description: Test cases for Q.NO.20**

**Assigned things:**

**1.pen 2.paper**

**3.question paper**

**Enter the two things that are placed on the shared table: 1**

**2**

**Output: 1st Student completed the task Assigned things:**

**1.pen 2.paper**

**3.question paper**

**Enter the two things that are placed on the shared table: 2**

**3**

**Output: 2nd Student completed the task**

**Assigned things:**

**1.pen 2.paper**

1. **question paper**

**Enter the two things that are placed on the shared table: 1**

**3**

**Output: 3rd Student completed the task.**

* + **After completing all the students task it will print “all the studets have completed the task”**
  + **If we enter wrong inputs then it will print “ enter the correct inputs” Assigned things:1.pen**

**2.paper 3.question paper**

**Enter the two things that are placed on the shared table: 3 4**

**Output: Enter correct inputs**

**Test cases for Q.NO.7**

**Enter the no,of processes you want:**

**5**

**Enter any data naresh**

**Response time of Process is 0 (m\_sec) :**

**Response time of Process is 1 (m\_sec) :**

**Response time of Process is 2 (m\_sec) :**

**Response time of Process is 3 (m\_sec) :**

**Response time of Process is 4 (m\_sec) :**

**Process Number Response Time ProcessType Priority**

1. **4202832ms Non-Interactive Low priority--**
2. **0ms Interactive High proiority**
3. **30ms Interactive High proiority**
4. **0ms Interactive High proiority**
5. **11473744ms Non-Interactive Low priority--**
   * **Have you made minimum 5 revisions of solution on GitHub?**