IARE

EDUCATION FOR LIBERATION

IARE
INSTITUTE OF
AERONAUTICAL ENGINEERING
(An Autonomous Institute affiliated to JNTUH, Hyderabad)
Dundigal, Hyderabad - 500 043

LABORATORY WORK BOOK

Name of the Student : N. Ravi Chandrika

Class CSD-B Semester IIIrd Semester

Course Code : ACSD10 Course Name : OS laboratory

Roll Number									
2	3	9	5	1	A	6	7	B	3

Name of the Course Faculty Ms. G. Indu Faculty ID : IARE10971

Exercise Number. 2 Week Number : 2 Date : 06/09/24

S. No.	Exercise Number	EXERCISE NAME	MARKS AWARDED						
			Aim/ Preparation	Algorithm / Procedure Performance in the Lab		Source Code Calculations and Graphs	Program Execution Results and Error Analysis	Viva - Voce	Total
			4	4		4	4	4	20
1	2.1	Managing system & user processes	4	2	2	4	4	4	20
2	2.2	Managing Jobscheduling in a computing centre	4	2	2	4	4	4	20
3	2.3	Managing print Jobs	4	2	2	4	4	4	20
4	2.4	Task scheduling in a multi-user system	4	2	2	4	4	4	20
5	2.5	Job scheduling in a computing cluster	4	2	2	4	4	4	20
6									
7									
8									
9									
10									
11									
12									

N. Ravichandrika
Signature of the Student

Signature of the Faculty

2.1 **Aim:** You are managing a computer system that runs various types of processes, categorized into system processes and user processes. System processes critical for functioning of OS and are given higher priority compared to user processes.

Code:

```
from queue import Queue
def calculate_total_time(system_processes, user_processes):
    process_queue = Queue()
    for process, time in system_processes.items():
        process_queue.put((process, time))
    for process, time in user_processes.items():
        process_queue.put((process, time))
    total_time = 0
    while not process_queue.empty():
        process, time = process_queue.get()
        total_time += time
    return total_time
system_processes = {'Process A': 5, 'Process B': 3, 'Process C': 7}
user_processes = {'Process D': 4, 'Process E': 2, 'Process F': 6}
total_time = calculate_total_time(system_processes, user_processes)
print(f"The total time required to complete all processes is:
                    {total_time} units").
```

Output: The total time required to complete all processes is: 27 units.

2.2 Aim: You are managing a computing centre that processes jobs submitted by various departments of an organization. Jobs are categorized into two priority levels: high priority and the low priority.

Code:

```
from queue import Queue
def calculate-total-time(high-priority-jobs, low-priority-jobs):
    job-queue = Queue()
    for job, time in high-priority-jobs.items():
        job-queue.put((job,time))
    for job, time in low-priority-jobs.items():
        job-queue.put((job,time))
    total-time = 0
    while not job-queue.empty()
        job, time = job-queue.get()
        total-time += time
    return total-time
high-priority-jobs = {'Job A': 8, 'Job B': 5, 'Job C': 10}
low-priority-jobs = {'Job D': 6, 'Job E': 3, 'Job F': 7}
total-time = calculate-total-time(high-priority-jobs, low-priority-jobs
print(f"The total time required to complete all jobs is: {total-time}, units"
```

Output: The total time required to complete all jobs is:
39 units.

Q.3 Aim: You are managing a shared printing environment where the point jobs differ from depaatment todepartment. Point jobs are categorized into high priority & Low priority.

Code:

```
from queue import Queue
def calculate-total-time(high-poiority-jobs, low-priority-jobs):
    job-queue = Queue()
    for job, time in high-poiority-jobs.items():
        job_queue.put((job,time))
    for job, time in low-poiority-jobs.items():
        job-queue.put((job, time))
    total-time = 0
    while not job-Queue.empty():
        job, time = job-queue.get()
        total-time += time
    return total-time
high-priority-jobs = {'Job A':15, 'Job B':10, 'Job C' : 20}
low-priority-jobs = {'Job D' : 5, 'Job E' : 8, 'Job F' : 3}
total-time = calculate-total-time(high-poiority-jobs, low_priority-
                                  jobs)
Point(f"The total time required to complete all jobs is: {total-time}
                                  units")
```

Output: The total time required to complete all print jobs is: 61 units.

Q-4 **Aim:** You are managing a multi-user system where tasks from different users need to be scheduled for processing. Tasks are categorized into two priority levels: high priority and low poriority.

Code:

```
 from queue import Queue
 def calculate-total-time(high-poriority-jobs, low-poriority-jobs):
     task_queue = Queue()
     for task, time in high-priority-tasks.items():
         task-queue.put((task, time))
     for task, time in low-poriority-tasks.items():
         task-queue.put((task, time))
     total-time = 0
     while not task-queue.empty():
         -, time = task-queue.get()
         total-time += time
     return total-time
high-priority-jobs = {'Task A' : 12, 'Task B' : 8, 'Task C' : 15}
low-priority-jobs = {'Task D' : 6, 'Task E' : 4, 'Task F' : 10}
total-time = calculate-total-time(high-priority-jobs, low-priority
                                              jobs)
print(f"The time required to complete all jobs is: {total-tim
                                              units")
```

Output: The total time required to complete all jobs is:
55 units.

Q.5 Affect you are managing a computing cluster that process jobs submitted by departments of a research institution. Jobs are categorised into : high priority, medium priority & the low priority.

Code:

```
From queue import Queue
def calculate_total_time( high_priority_jobs, medium, low):
    job_queue = Queue()
    for job, time in high_priority_jobs.items():
        job_queue.put((job, time))
    for job , time in medium_priority_jobs.items():
        job_queue.put((job, time))
    for job, time in low_priority_jobs.items():
        job_queue.put((job, time)).
    total_time = 0
    While not job_queue.empty():
        _, time = job_queue.get()
        total_time + = time
    return total_time
high_priority_jobs = {'Job A' : 20, 'Job B' : 15, 'Job C' : 25}
Medium_priority_jobs = {'Job D' : 10 , 'Job E' : 12, 'Job F' : 8},
low_priority_jobs = {'Job G' : 5, 'Job H' : 4, 'Job I' : 6}
total_time = calculate_total_time (high_priority_jobs,
                          medium_priority_jobs, low_priority_jobs)
```

ROLL NUMBER :

```
print(f"The total time required to complete all the tasks
is : {total_time} units")
```

Output: The total time required to complete all the tasks is :
105 units.