K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A2 **Roll No.: 1911027**

Experiment No. 03

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE: Implementation of Basic Process management algorithms – Non Pre-emptive (FCFS , SJF, priority)

AIM: To implement basic Non –Pre-emptive Process management algorithms (FCFS , SJF , Priority)

Expected Outcome of Experiment:

CO 2. To understand the concept of process, thread and resource management.

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems" McGraw Hill Third Edition.**
3. **William Stallings, "Operating System Internal & Design Principles", Pearson.**
4. **Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall.**

Pre Lab/ Prior Concepts:

Most systems have a large number of processes with short CPU bursts interspersed between I/O requests and a small number of processes with long CPU bursts. To provide good time-sharing performance, we may preempt a running process to let

another one run. The ready list, also known as a run queue, in the operating system keeps a list of all processes that are ready to run and not blocked on some I/O or other system request, such as a semaphore. Then entries in this list are pointers to the process control block, which stores all information and state about a process.

When an I/O request for a process is complete, the process moves from the *waiting* state to the *ready* state and gets placed on the run queue.

The process scheduler is the component of the operating system that is responsible for deciding whether the currently running process should continue running and, if not, which process should run next. There are four events that may occur where the scheduler needs to step in and make this decision:

1. The current process goes from the *running* to the *waiting* state because it issues an I/O request or some operating system request that cannot be satisfied immediately.

2. The current process terminates.

3. A timer interrupt causes the scheduler to run and decide that a process has run for its allotted interval of time and it is time to move it from the *running* to the *ready* state.

4. An I/O operation is complete for a process that requested it and the process now moves from the *waiting* to the*ready* state. The scheduler may then decide to preempt the currently-running process and move this *ready* process into the *running* state.

The decisions that the scheduler makes concerning the sequence and length of time that processes may run is called the scheduling algorithm (or scheduling policy). These decisions are not easy ones, as the scheduler has only a limited amount of information about the processes that are ready to run. A good scheduling algorithm should:

1. Be fair – give each process a fair share of the CPU, allow each process to run in a reasonable amount of time.
2. Be efficient – keep the CPU busy all the time.

3. Maximize throughput – service the largest possible number of jobs in a given amount of time; minimize the amount of time users must wait for their results.
4. Minimize response time – interactive users should see good performance
5. Minimize overhead – don't waste too many resources. Keep scheduling time and context switch time at a minimum.
6. Maximize resource use – favor processes that will use underutilized resources. There are two motives for this. Most devices are slow compared to CPU operations. We'll achieve better system throughput by keeping devices busy as often as possible. The second reason is that a process may be holding a key resource and other, possibly more important, processes cannot use it until it is released. Giving the process more CPU time may free up the resource quicker.
7. Avoid indefinite postponement – every process should get a chance to run eventually.

Description of the application to be implemented:

First-Come, First-Served Scheduling:

First come first serve (FCFS) scheduling algorithm simply schedules the jobs according to their arrival time. The job which comes first in the ready queue will get the CPU first. The lesser the arrival time of the job, the sooner will the job get the CPU. FCFS scheduling may cause the problem of starvation if the burst time of the first process is the longest among all the jobs. As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue and, when the CPU becomes free, it should be assigned to the process at the beginning of the queue.

Characteristics of FCFS method:

- It supports non-preemptive and pre-emptive scheduling algorithm.
- Jobs are always executed on a first-come, first-serve basis.
- It is easy to implement and use.
- This method is poor in performance, and the general wait time is quite high.

Advantages of FCFS:

- The simplest form of a CPU scheduling algorithm
- Easy to program

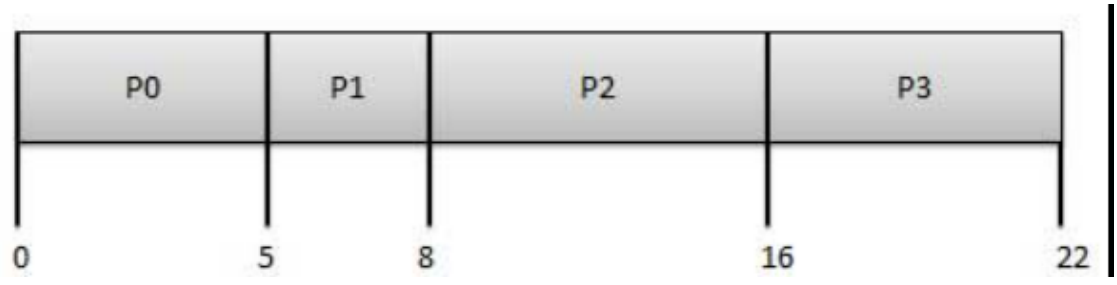
Disadvantages of FCFS:

- It is a Non-Preemptive CPU scheduling algorithm, so after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.
- The Average Waiting Time is high.
- Short processes that are at the back of the queue have to wait for the long process at the front to finish.
- Not an ideal technique for time-sharing systems.
- Because of its simplicity, FCFS is not very efficient.

Example of FCFS scheduling: A real-life example of the FCFS method is buying a movie ticket on the ticket counter. In this scheduling algorithm, a person is served according to the queue manner. The person who arrives first in the queue first buys the ticket and then the next one. This will continue until the last person in the queue purchases the ticket. Using this algorithm, the CPU process works in a similar manner.

Process	Arrival Time	Execute Time
P0	0	5
P1	1	3
P2	2	8
P3	3	6

Gantt chart:



Implementation details: (printout of code)

CODE:

```
def sort(process):
    for i in process:
        for j in range(0,len(process)-1):
            if(int(process[j][1])>int(process[j+1][1])):
                temp=process[j]
                process[j]=process[j+1]
                process[j+1]=temp

def checkarrival(process,processtaken,time):
    temp=list()
    for i in range(0,len(process)):
        if(int(process[i][1])==time and process[i][0] not in processtaken
):
            temp.append(process[i])
    return temp

print("------------------------------------------------------------------
------------------------------------------------------------------")
print("------------------------------------------------------------------
FCFS----------------------------------------------------------------")
print("------------------------------------------------------------------
------------------------------------------------------------------")
n=int(input("Enter number of processes : "))
print("------------------------------------------------------------------
------------------------------------------------------------------")
process=list()
for i in range(0,n):
    inp=input("Enter PID, Arrival Time, Burst Time : ").split()
    process.append(inp)
totaltime=0
sort(process)
dic=dict()
for i in process:
    dic[i[0]]=i[1]
print("------------------------------------------------------------------
------------------------------------------------------------------")
k=1
```

```
totaltime=totaltime+int(process[0][1])+int(process[0][2])
flag=0
while(flag!=1):
    if(totaltime>=int(process[k][1])):
        temptime=0
        for i in range(k,n):
            if(int(process[i][1])<=totaltime):
                k=k+1
                temptime=temptime+int(process[i][2])
                if(k==n):
                    flag=1
        totaltime=totaltime+temptime
    else:
        totaltime=totaltime+1
processtaken=list()
readyqueue=list()
processing=list()
waiting=0
turnaround=0
waitinglist=list()
turnaroundlist=list()
waitinglist.append(int(process[0][1]))
turnaroundlist.append(int(process[0][2]))
duration=list()
print("Ready queue instances : ")
for i in range(int(process[0][1]),totaltime+1):
    print("Time : ",i,end=" ")
    pro=checkarrival(process,processtaken,i)
    for j in pro:
        if(j not in processtaken):
            readyqueue.append(j)
            processtaken.append(j)
    print("Ready queue : ",readyqueue,end=" ")
    if(len(processing)==0):
        if(len(readyqueue)>0):
            processing.append(readyqueue[0])
            print("Processing queue : ",processing,end=" ")
            readyqueue.pop(0)
    else:
        if(int(processing[0][2])==0):
            duration.append(processing[0][0])
```

```
            processing.pop()
            if(len(readyqueue)>0):
                processing.append(readyqueue[0])
                print("Processing queue : ",processing,end=" ")
                readyqueue.pop(0)
        else:
            processing[0][2]=int(processing[0][2])-1
            print("Processing queue : ",processing,end=" ")
            duration.append(processing[0][0])
            if(int(processing[0][2])==0):
                processing.pop()
                if(len(readyqueue)>0):
                    processing.append(readyqueue[0])
                    print("Processing queue : ",processing,end=" ")
                    readyqueue.pop(0)
    if(len(pro)==0 and len(processtaken)!=len(process) and len(processing
)==0):
        duration.append("X")
    print()
print("-------------------------------------------------------------------
-------------------------------------------------------------------")
temp=list()
t=0
temp2=list()
for i in range(0,len(duration)-1):
    t=t+1
    flag=0
    if(duration[i]!=duration[i+1]):
        temp2=[duration[i],t]
        temp.append(temp2)
        flag=1
    if(i==len(duration)-2):
        if(flag==1):
            temp2=[duration[i+1],t+1]
            temp.append(temp2)
        else:
            temp2=[duration[i],t+1]
            temp.append(temp2)
if(int(process[0][1])>0):
    for i in temp:
        i[1]=i[1]+int(process[0][1])
```

```
print("Gantt Chart  : ")
start=int(process[0][1])
forcalculation=list()
for i in temp:
    lst=list()
    lst=[None]*3
    if(i[0]!="X"):
        print("(",start,") P",i[0],"(",i[1],")",end=" ")
        lst[0]=start
        lst[1]=i[0]
        lst[2]=i[1]
        forcalculation.append(lst)
        for j in range(start,i[1]):
            print("  ",end=" ")
    start=i[1]
print("\n------------------------------------------------------------------------
-------------------------------------------------------------------------")
print("Waiting time of processes : ")
for i in range(0,len(process)):
    if(i==0):
        print("P",forcalculation[i][1]," : 0")
    else:
        print("P",forcalculation[i][1]," : ",forcalculation[i][0]-
int(dic[forcalculation[i][1]]))
        waiting=waiting+forcalculation[i][0]-
int(dic[forcalculation[i][1]])
print("------------------------------------------------------------------------
-------------------------------------------------------------------------")
print("Average waiting time  : ",waiting/len(process))
print("------------------------------------------------------------------------
-------------------------------------------------------------------------")
print("Turnaround time of processes : ")
for i in range(0,len(process)):
    print("P",forcalculation[i][1]," : ",forcalculation[i][2]-
int(dic[forcalculation[i][1]]))
    turnaround=turnaround+forcalculation[i][2]-
int(dic[forcalculation[i][1]])
print("------------------------------------------------------------------------
-------------------------------------------------------------------------")
print("Average turnaround time : ",turnaround/len(process))
```

```
print("------------------------------------------------------------------------
------------------------------------------------------------------------")
```

OUTPUT:

```
--------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------FCFS----------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------
Enter number of processes : 3
--------------------------------------------------------------------------------------------------------------------------------
Enter PID, Arrival Time, Burst Time : 1 2 3
Enter PID, Arrival Time, Burst Time : 2 0 1
Enter PID, Arrival Time, Burst Time : 3 4 2
--------------------------------------------------------------------------------------------------------------------------------
Ready queue instances :
Time :  0 Ready queue :  [['2', '0', '1']] Processing queue :  [['2', '0', '1']]
Time :  1 Ready queue :  [] Processing queue :  [['2', '0', 0]]
Time :  2 Ready queue :  [['1', '2', '3']] Processing queue :  [['1', '2', '3']]
Time :  3 Ready queue :  [] Processing queue :  [['1', '2', 2]]
Time :  4 Ready queue :  [['3', '4', '2']] Processing queue :  [['1', '2', 1]]
Time :  5 Ready queue :  [['3', '4', '2']] Processing queue :  [['1', '2', 0]] Processing queue :  [['3', '4', '2']]
Time :  6 Ready queue :  [] Processing queue :  [['3', '4', 1]]
Time :  7 Ready queue :  [] Processing queue :  [['3', '4', 0]]
--------------------------------------------------------------------------------------------------------------------------------
Gantt Chart  :
( 0 ) P 2 ( 1 )   ( 2 ) P 1 ( 5 )       ( 5 ) P 3 ( 7 )
--------------------------------------------------------------------------------------------------------------------------------
Waiting time of processes :
P 2  : 0
P 1  :  0
P 3  :  1
--------------------------------------------------------------------------------------------------------------------------------
Average waiting time  :  0.3333333333333333
--------------------------------------------------------------------------------------------------------------------------------
Turnaround time of processes :
P 2  :  1
P 1  :  3
P 3  :  3
--------------------------------------------------------------------------------------------------------------------------------
Average turnaround time :  2.3333333333333335
--------------------------------------------------------------------------------------------------------------------------------
```

Shortest job first :

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive. It significantly reduces the average waiting time for other processes awaiting execution. The full form of SJF is Shortest Job First. It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available. It is a Greedy Algorithm. It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Characteristics of SJF Scheduling:

- It is associated with each job as a unit of time to complete.
- This algorithm method is helpful for batch-type processing, where waiting for jobs to complete is not critical.
- It can improve process throughput by making sure that shorter jobs are executed first, hence possibly have a short turnaround time.
- It improves job output by offering shorter jobs, which should be executed first, which mostly have a shorter turnaround time.

Advantages of SJF:

- SJF is frequently used for long term scheduling.
- It reduces the average waiting time over FCFS (First Come First Serve) algorithm.
- SJF method gives the lowest average waiting time for a specific set of processes.
- It is appropriate for the jobs running in batch, where run times are known in advance.

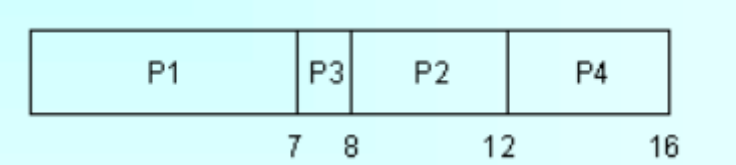
Disadvantages/Cons of SJF:

- Job completion time must be known earlier, but it is hard to predict.
- It is often used in a batch system for long term scheduling.
- SJF can't be implemented for CPU scheduling for the short term. It is because there is no specific method to predict the length of the upcoming CPU burst.
- This algorithm may cause very long turnaround times or starvation.

Example:

Process:	p1	p2	p3	p4
Arrival time:	0	2	4	5
Burst time:	7	4	1	4

Gantt chart:



<u>Implementation details:</u> (printout of code)

CODE:

```
def sort(process):
    for i in process:
        for j in range(0,len(process)-1):
            if(int(process[j][1])>int(process[j+1][1])):
                temp=process[j]
                process[j]=process[j+1]
                process[j+1]=temp

def checkarrival(process,processtaken,time):
```

```
    temp=list()
    for i in range(0,len(process)):
        if(int(process[i][1])==time and process[i][0] not in processtaken
):
            temp.append(process[i])
    return temp

def sjf(readyqueue):
    for i in readyqueue:
        for j in range(0,len(readyqueue)-1):
            if(int(readyqueue[j][2])>int(readyqueue[j+1][2])):
                temp=readyqueue[j]
                readyqueue[j]=readyqueue[j+1]
                readyqueue[j+1]=temp
    return readyqueue

print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
print("----------------------------------------------------------------------
-------SJF-------------------------------------------------------------------
---------------")
print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
n=int(input("Enter number of processes : "))
print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
process=list()
for i in range(0,n):
    inp=input("Enter PID, Arrival Time, Burst Time : ").split()
    process.append(inp)
totaltime=0
sort(process)
dic=dict()
for i in process:
    dic[i[0]]=i[1]
print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
```

```
k=1
totaltime=totaltime+int(process[0][1])+int(process[0][2])
flag=0
while(flag!=1):
    if(totaltime>=int(process[k][1])):
        temptime=0
        for i in range(k,n):
            if(int(process[i][1])<=totaltime):
                k=k+1
                temptime=temptime+int(process[i][2])
                if(k==n):
                    flag=1
        totaltime=totaltime+temptime
    else:
        totaltime=totaltime+1
processtaken=list()
readyqueue=list()
processing=list()
waiting=0
turnaround=0
waitinglist=list()
turnaroundlist=list()
waitinglist.append(int(process[0][1]))
turnaroundlist.append(int(process[0][2]))
duration=list()
finalprocesssequence=list()
print("Ready queue instances : ")
for i in range(int(process[0][1]),totaltime+1):
    print("Time : ",i,end=" ")
    pro=checkarrival(process,processtaken,i)
    for j in pro:
        if(j not in processtaken):
            readyqueue.append(j)
            processtaken.append(j)
    if(len(readyqueue)>1):
        readyqueue=sjf(readyqueue)
    print("Ready queue : ",readyqueue,end=" ")
    if(len(processing)==0):
        if(len(readyqueue)>0):
            processing.append(readyqueue[0])
            print("Processing queue : ",processing,end=" ")
```

```
            finalprocesssequence.append(readyqueue[0])
            readyqueue.pop(0)
    else:
        processing[0][2]=int(processing[0][2])-1
        print("Processing queue : ",processing,end=" ")
        duration.append(processing[0][0])
        if(int(processing[0][2])==0):
            processing.pop()
            if(len(readyqueue)>0):
                processing.append(readyqueue[0])
                print("Processing queue : ",processing,end=" ")
                finalprocesssequence.append(readyqueue[0])
                readyqueue.pop(0)
    if(len(pro)==0 and len(processtaken)!=len(process) and len(processing
)==0):
        duration.append("X")
    print()
print("----------------------------------------------------------------
----------------------------------------------------------------
---------------")
temp=list()
t=0
temp2=list()
for i in range(0,len(duration)-1):
    t=t+1
    flag=0
    if(duration[i]!=duration[i+1]):
        temp2=[duration[i],t]
        temp.append(temp2)
        flag=1
    if(i==len(duration)-2):
        if(flag==1):
            temp2=[duration[i],t+1]
            temp.append(temp2)
        else:
            temp2=[duration[i+1],t+1]
            temp.append(temp2)
if(int(process[0][1])>0):
    for i in temp:
        i[1]=i[1]+int(process[0][1])
print("Gantt Chart  : ")
```

```
start=int(process[0][1])
forcalculation=list()
for i in temp:
    lst=list()
    lst=[None]*3
    if(i[0]!="X"):
        print("(",start,") P",i[0],"(",i[1],")",end=" ")
        lst[0]=start
        lst[1]=i[0]
        lst[2]=i[1]
        forcalculation.append(lst)
        for j in range(start,i[1]):
            print("  ",end=" ")
    start=i[1]
print("\n----------------------------------------------------------------------
----------------------------------------------------------------------
-----------------")
print("Waiting time of processes : ")
for i in range(0,len(finalprocesssequence)):
    if(i==0):
        print("P",forcalculation[i][1]," : 0")
    else:
        print("P",forcalculation[i][1]," : ",forcalculation[i][0]-
int(dic[forcalculation[i][1]]))
        waiting=waiting+forcalculation[i][0]-
int(dic[forcalculation[i][1]])
print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
print("Average waiting time  : ",waiting/len(process))
print("----------------------------------------------------------------------
----------------------------------------------------------------------
---------------")
print("Turnaround time of processes : ")
for i in range(0,len(finalprocesssequence)):
    print("P",forcalculation[i][1]," : ",forcalculation[i][2]-
int(dic[forcalculation[i][1]]))
    turnaround=turnaround+forcalculation[i][2]-
int(dic[forcalculation[i][1]])
```

```
print("--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------")
print("Average turnaround time : ",turnaround/len(process))
print("--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------")
```

OUTPUT:

```
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------SJF-------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------
Enter number of processes : 4
--------------------------------------------------------------------------------------------------------------------------------------------------------
Enter PID, Arrival Time, Burst Time : 0 0 4
Enter PID, Arrival Time, Burst Time : 1 2 1
Enter PID, Arrival Time, Burst Time : 2 1 2
Enter PID, Arrival Time, Burst Time : 3 4 3
--------------------------------------------------------------------------------------------------------------------------------------------------------
Ready queue instances :
Time :  0 Ready queue :  [['0', '0', '4']] Processing queue :  [['0', '0', '4']]
Time :  1 Ready queue :  [['2', '1', '2']] Processing queue :  [['0', '0', 3]]
Time :  2 Ready queue :  [['1', '2', '1'], ['2', '1', '2']] Processing queue :  [['0', '0', 2]]
Time :  3 Ready queue :  [['1', '2', '1'], ['2', '1', '2']] Processing queue :  [['0', '0', 1]]
Time :  4 Ready queue :  [['1', '2', '1'], ['2', '1', '2'], ['3', '4', '3']] Processing queue :  [['0', '0', 0]] Processing queue :  [['1', '2', '1']]
Time :  5 Ready queue :  [['2', '1', '2'], ['3', '4', '3']] Processing queue :  [['1', '2', 0]] Processing queue :  [['2', '1', '2']]
Time :  6 Ready queue :  [['3', '4', '3']] Processing queue :  [['2', '1', 1]]
Time :  7 Ready queue :  [['3', '4', '3']] Processing queue :  [['2', '1', 0]] Processing queue :  [['3', '4', '3']]
Time :  8 Ready queue :  [] Processing queue :  [['3', '4', 2]]
Time :  9 Ready queue :  [] Processing queue :  [['3', '4', 1]]
Time :  10 Ready queue :  [] Processing queue :  [['3', '4', 0]]
--------------------------------------------------------------------------------------------------------------------------------------------------------
Gantt Chart  :
( 0 ) P 0 ( 4 )          ( 4 ) P 1 ( 5 )   ( 5 ) P 2 ( 7 )     ( 7 ) P 3 ( 10 )
--------------------------------------------------------------------------------------------------------------------------------------------------------
Waiting time of processes :
P 0  : 0
P 1  : 2
P 2  : 4
P 3  : 3
--------------------------------------------------------------------------------------------------------------------------------------------------------
Average waiting time  : 2.25
--------------------------------------------------------------------------------------------------------------------------------------------------------
Turnaround time of processes :
P 0  : 4
P 1  : 3
P 2  : 6
P 3  : 6
--------------------------------------------------------------------------------------------------------------------------------------------------------
Average turnaround time :  4.75
--------------------------------------------------------------------------------------------------------------------------------------------------------
```

Priority scheduling :

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority should be carried out first, whereas jobs with equal priorities are carried out on a round-robin or FCFS basis. Priority depends upon memory requirements, time requirements, etc. In non preemptive type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That's because it doesn't need special hardware (for example, a timer) like preemptive scheduling.

Characteristics of Priority Scheduling:

- A CPU algorithm that schedules processes based on priority.
- It used in Operating systems for performing batch processes.
- If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
- In priority scheduling, a number is assigned to each process that indicates its priority level.
- Lower the number, higher is the priority.
- In this type of scheduling algorithm, if a newer process arrives, that is having a higher priority than the currently running process, then the currently running process is preempted.

Advantages of priority scheduling:

- Easy to use scheduling method
- Processes are executed on the basis of priority so high priority does not need to wait for long which saves time
- This method provides a good mechanism where the relative important of each process may be precisely defined.
- Suitable for applications with fluctuating time and resource requirements.

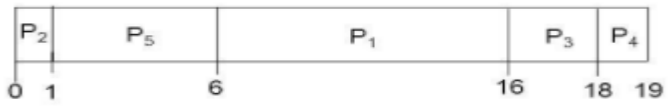
Disadvantages of priority scheduling:

- If the system eventually crashes, all low priority processes get lost.
- If high priority processes take lots of CPU time, then the lower priority processes may starve and will be postponed for an indefinite time.
- This scheduling algorithm may leave some low priority processes waiting indefinitely.
- A process will be blocked when it is ready to run but has to wait for the CPU because some other process is running currently.

Example:

Process	Burst Time	Priority	arrival time
P_1	10	3	0
P_2	1	1	0
P_3	2	4	0
P_4	1	5	0
P_5	5	2	0

Gantt chart:



Implementation details: (printout of code)

CODE:

```
def sort(process):
    for i in process:
        for j in range(0,len(process)-1):
            if(int(process[j][1])>int(process[j+1][1])):
                temp=process[j]
                process[j]=process[j+1]
                process[j+1]=temp

def checkarrival(process,processtaken,time):
    temp=list()
```

```
    for i in range(0,len(process)):
        if(int(process[i][1])==time and process[i][0] not in processtaken
):
            temp.append(process[i])
    return temp

def priority(readyqueue):
    for i in readyqueue:
        for j in range(0,len(readyqueue)-1):
            if(int(readyqueue[j][3])>int(readyqueue[j+1][3])):
                temp=readyqueue[j]
                readyqueue[j]=readyqueue[j+1]
                readyqueue[j+1]=temp
    return readyqueue

print("------------------------------------------------------------------
------------------------------------------------------------------------
---------------")
print("------------------------------------------------------------------
-----PRIORITY----------------------------------------------------------
---------------")
print("------------------------------------------------------------------
------------------------------------------------------------------------
---------------")
n=int(input("Enter number of processes : "))
print("------------------------------------------------------------------
------------------------------------------------------------------------
---------------")
process=list()
for i in range(0,n):
    inp=input("Enter PID, Arrival Time, Burst Time, priority : ").split()
    process.append(inp)
totaltime=0
sort(process)
dic=dict()
for i in process:
    dic[i[0]]=i[1]
print("------------------------------------------------------------------
------------------------------------------------------------------------
---------------")
k=1
```

```
totaltime=totaltime+int(process[0][1])+int(process[0][2])
flag=0
while(flag!=1):
    if(totaltime>=int(process[k][1])):
        temptime=0
        for i in range(k,n):
            if(int(process[i][1])<=totaltime):
                k=k+1
                temptime=temptime+int(process[i][2])
                if(k==n):
                    flag=1
        totaltime=totaltime+temptime
    else:
        totaltime=totaltime+1
processtaken=list()
readyqueue=list()
processing=list()
waiting=0
turnaround=0
waitinglist=list()
turnaroundlist=list()
waitinglist.append(int(process[0][1]))
turnaroundlist.append(int(process[0][2]))
duration=list()
finalprocesssequence=list()
print("Ready queue instances : ")
for i in range(int(process[0][1]),totaltime+1):
    print("Time : ",i,end=" ")
    pro=checkarrival(process,processtaken,i)
    for j in pro:
        if(j not in processtaken):
            readyqueue.append(j)
            processtaken.append(j)
    if(len(readyqueue)>1):
        readyqueue=priority(readyqueue)
    print("Ready queue : ",readyqueue,end=" ")
    if(len(processing)==0):
        if(len(readyqueue)>0):
            processing.append(readyqueue[0])
            print("Processing queue : ",processing,end=" ")
            finalprocesssequence.append(readyqueue[0])
```

```
            readyqueue.pop(0)
    else:
        processing[0][2]=int(processing[0][2])-1
        print("Processing queue : ",processing,end=" ")
        duration.append(processing[0][0])
        if(int(processing[0][2])==0):
            processing.pop()
            if(len(readyqueue)>0):
                processing.append(readyqueue[0])
                print("Processing queue : ",processing,end=" ")
                finalprocesssequence.append(readyqueue[0])
                readyqueue.pop(0)
    if(len(pro)==0 and len(processtaken)!=len(process) and len(processing
)==0):
        duration.append("X")
    print()
print("----------------------------------------------------------------------
----------------------------------------------------------------------------
---------------")
temp=list()
t=0
temp2=list()
for i in range(0,len(duration)-1):
    t=t+1
    flag=0
    if(duration[i]!=duration[i+1]):
        temp2=[duration[i],t]
        temp.append(temp2)
        flag=1
    if(i==len(duration)-2):
        if(flag==1):
            temp2=[duration[i+1],t+1]
            temp.append(temp2)
        else:
            temp2=[duration[i],t+1]
            temp.append(temp2)
if(int(process[0][1])>0):
    for i in temp:
        i[1]=i[1]+int(process[0][1])
print("Gantt Chart  : ")
start=int(process[0][1])
```

```
forcalculation=list()
for i in temp:
    lst=list()
    lst=[None]*3
    if(i[0]!="X"):
        print("(",start,") P",i[0],"(",i[1],")",end=" ")
        lst[0]=start
        lst[1]=i[0]
        lst[2]=i[1]
        forcalculation.append(lst)
        for j in range(start,i[1]):
            print("  ",end=" ")
    start=i[1]
print("\n----------------------------------------------------------------
----------------------------------------------------------------------
-----------------")
print("Waiting time of processes : ")
for i in range(0,len(finalprocesssequence)):
    if(i==0):
        print("P",forcalculation[i][1]," : 0")
    else:
        print("P",forcalculation[i][1]," : ",forcalculation[i][0]-
int(dic[forcalculation[i][1]]))
        waiting=waiting+forcalculation[i][0]-
int(dic[forcalculation[i][1]])
print("----------------------------------------------------------------
----------------------------------------------------------------------
--------------")
print("Average waiting time  : ",waiting/len(process))
print("----------------------------------------------------------------
----------------------------------------------------------------------
--------------")
print("Turnaround time of processes : ")
for i in range(0,len(finalprocesssequence)):
    print("P",forcalculation[i][1]," : ",forcalculation[i][2]-
int(dic[forcalculation[i][1]]))
    turnaround=turnaround+forcalculation[i][2]-
int(dic[forcalculation[i][1]])
print("----------------------------------------------------------------
----------------------------------------------------------------------
--------------")
```

```
print("Average turnaround time : ",turnaround/len(process))
print("--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
---------------")
```

OUTPUT:

```
--------------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------PRIORITY--------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------------------------------------------
Enter number of processes : 4
--------------------------------------------------------------------------------------------------------------------------------------------------------
Enter PID, Arrival Time, Burst Time, priority : 0 2 7 3
Enter PID, Arrival Time, Burst Time, priority : 1 1 3 0
Enter PID, Arrival Time, Burst Time, priority : 2 0 1 1
Enter PID, Arrival Time, Burst Time, priority : 3 3 2 2
--------------------------------------------------------------------------------------------------------------------------------------------------------
Ready queue instances :
Time :  0 Ready queue :  [['2', '0', '1', '1']] Processing queue :  [['2', '0', '1', '1']]
Time :  1 Ready queue :  [['1', '1', '3', '0']] Processing queue :  [['2', '0', 0, '1']] Processing queue :  [['1', '1', '3', '0']]
Time :  2 Ready queue :  [['0', '2', '7', '3']] Processing queue :  [['1', '1', 2, '0']]
Time :  3 Ready queue :  [['3', '3', '2', '2'], ['0', '2', '7', '3']] Processing queue :  [['1', '1', 1, '0']]
Time :  4 Ready queue :  [['3', '3', '2', '2'], ['0', '2', '7', '3']] Processing queue :  [['1', '1', 0, '0']] Processing queue :  [['3', '3', '2', '2']]
Time :  5 Ready queue :  [['0', '2', '7', '3']] Processing queue :  [['3', '3', 1, '2']]
Time :  6 Ready queue :  [['0', '2', '7', '3']] Processing queue :  [['3', '3', 0, '2']] Processing queue :  [['0', '2', '7', '3']]
Time :  7 Ready queue :  [] Processing queue :  [['0', '2', 6, '3']]
Time :  8 Ready queue :  [] Processing queue :  [['0', '2', 5, '3']]
Time :  9 Ready queue :  [] Processing queue :  [['0', '2', 4, '3']]
Time :  10 Ready queue :  [] Processing queue :  [['0', '2', 3, '3']]
Time :  11 Ready queue :  [] Processing queue :  [['0', '2', 2, '3']]
Time :  12 Ready queue :  [] Processing queue :  [['0', '2', 1, '3']]
Time :  13 Ready queue :  [] Processing queue :  [['0', '2', 0, '3']]
--------------------------------------------------------------------------------------------------------------------------------------------------------
Gantt Chart  :
( 0 ) P 2 ( 1 )   ( 1 ) P 1 ( 4 )       ( 4 ) P 3 ( 6 )     ( 6 ) P 0 ( 13 )
--------------------------------------------------------------------------------------------------------------------------------------------------------
Waiting time of processes :
P 2  : 0
P 1  : 0
P 3  : 1
P 0  : 4
--------------------------------------------------------------------------------------------------------------------------------------------------------
Average waiting time  : 1.25
--------------------------------------------------------------------------------------------------------------------------------------------------------
Turnaround time of processes :
P 2  : 1
P 1  : 3
P 3  : 3
P 0  : 11
--------------------------------------------------------------------------------------------------------------------------------------------------------
Average turnaround time :  4.5
--------------------------------------------------------------------------------------------------------------------------------------------------------
```

Conclusion: Understood the concept of CPU scheduling and also seen some of the non preemptive algorithms like FCFS, SJF and priority scheduling. Understood how the processes are scheduled using all the algorithms. Implemented all the algorithms using python programming language.

Post Lab Objective Questions

1. What is the ready state of a process?
a) when process is scheduled to run after some execution
b) when process is unable to run until some task has been completed
c) when process is using the CPU
d) none of the mentioned

Ans: a) when process is scheduled to run after some execution

2. A process stack does not contain
a) function parameters
b) local variables
c) return addresses
d) PID of child process

Ans: d) PID of child process

3. A process can be terminated due to
a) normal exit
b) fatal error
c) killed by another process
d) all of the mentioned

Ans: d) all of the mentioned

Date: 30 / 9 / 2021 **Signature of faculty in-charge**