Experiment – 1

First Come First Serve

Introduction

First come first serve (FCFS) is a scheduling algorithm that executes queued requests in order of their arrival. It is easiest and simplest CPU scheduling algorithm. In this algorithm, processes which requests the CPU first gets the CPU first for it's entire execution time that is required to complete that process. It is a non-preemptive algorithm. It uses a queue data structure (FIFO) to maintain process execution order. Since context switches only occur upon process termination, and no reorganization of process execution order is required, hence overhead is minimal.

Algorithm

1. Implement a FIFO Queue to maintain process order
2. Add new processes to the queue as soon as they arrive
3. If CPU is idle, remove a process from the queue and provide it to the CPU
4. After the process is completed discard it and set the CPU idle till new process arrives

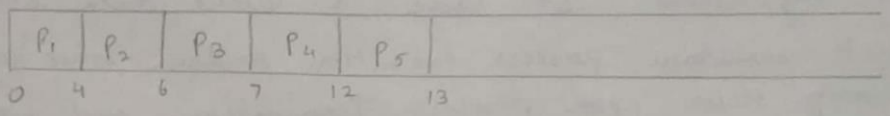
Numerical

Q. Conciden the following process whose arrival time and Burst time is given using FCFS algorithm

Process	Arrival Time	Burst Time
P_1	0	4
P_2	2	2
P_3	3	1
P_4	3	5
P_5	3	1

Ans

Process	Arrival Time	Burst Time	Completion time	TAT	WT	RT
P_1	0	4	4	4	0	0
P_2	2	2	6	4	2	2
P_3	3	1	7	4	3	3
P_4	3	5	12	9	4	4
P_5	3	1	13	10	9	9
		Average :	8.4	6.2	3.6	3.6

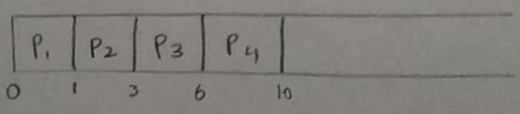


Q2. Concider the following processes whose arrival time & burst time is given, using First Come First Serve

Process	AT	BT
P_1	0	1
P_2	0	2
P_3	0	3
P_4	0	4

Ans

Process	AT	BT	CT	TAT	WT	RT
P_1	0	1	1	1	0	0
P_2	0	2	3	3	1	1
P_3	0	3	6	6	3	3
P_4	0	4	10	10	6	6
		Average	5	5	2.5	2.5

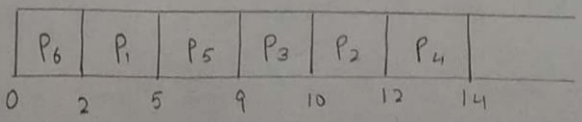


Q3. Concider the following process whose arrival time and burst time is given, using first come first serve

Process	Arrival Time (AT)	Burst Time (BT)
P_1	1	3
P_2	4	2
P_3	3	1
P_4	7	2
P_5	2	4
P_6	0	2

Ans

Process	AT	BT	CT	TAT	WT	RT
P_1	1	3	5	4	1	1
P_2	4	2	12	8	6	6
P_3	3	1	10	7	6	6
P_4	7	2	14	7	5	5
P_5	2	4	9	7	3	3
P_6	0	2	2	2	0	0
	Average		8.67	5.83	3.5	3.5



Code

```
class Process:
    def __init__(self, idx, AT, BT,) -> None:
        self.idx = idx
        self.AT = AT
        self.BT = BT
        self.CT = None
        self.firstExecution = None

    def calc(self):
        self.TAT = self.CT - self.AT
        self.WT = self.TAT - self.BT
        self.RT = self.firstExecution - self.AT

    def __repr__(self) -> str:
        return f"Process({self.idx}): {self.AT}, {self.BT}, {self.CT}, {self.TAT}
, {self.WT}, {self.RT}"

n = int(input("Number of processes: "))
arrivalTime = list(map(int, input("Arrival Times: ").split()))
burstTime = list(map(int, input("Burst Times: ").split()))

processes = sorted([Process(x+1,arrivalTime[x],burstTime[x]) for x in range(len(a
rrivalTime))],key=lambda x:x.AT)
cpuTime = processes[0].AT

for p in processes:
    if(p.AT <= cpuTime):
        p.firstExecution = cpuTime
    else:
        p.firstExecution = p.AT

    p.CT = p.firstExecution + p.BT
    p.calc()
    cpuTime = p.CT

print("\nProcess, AT, BT, CT, TAT, WT, RT")
[print(x) for x in sorted(processes,key=lambda x: x.idx)]

print("\nAverage:")
print(f"CT: {sum((x.CT for x in processes))/n}")
print(f"TAT: {sum((x.TAT for x in processes))/n}")
print(f"WT: {sum((x.WT for x in processes))/n}")
print(f"RT: {sum((x.RT for x in processes))/n}\n")
```

Output

```
PS D:\Drive\Sem 6\OS\lab> python -u "d:\Drive\Sem 6\OS\lab\fcfs.py"
Number of processes: 6
Arrival Times: 1 4 3 7 2 0
Burst Times: 3 2 1 2 4 2

Process, AT, BT, CT, TAT, WT, RT
Process(1): 1, 3, 5, 4, 1, 1
Process(2): 4, 2, 12, 8, 6, 6
Process(3): 3, 1, 10, 7, 6, 6
Process(4): 7, 2, 14, 7, 5, 5
Process(5): 2, 4, 9, 7, 3, 3
Process(6): 0, 2, 2, 2, 0, 0

Average:
CT: 8.666666666666666
TAT: 5.833333333333333
WT: 3.5
RT: 3.5

PS D:\Drive\Sem 6\OS\lab>
```