**COURSEWORK-ASSIGNMENT**

Name – Ashutosh Soni

Id – 2018ucp1505

Question: Write a report on

(1) new functions you have learned and a description of the function

(2) summary of what each program does

(3) Results and inferences from execution of the program

1. Process Intro (process-run.py)

Answer: This program allows us to see how the states of a process states are changing as it runs on a CPU. As we know that there are several states which process may take, they are:

* RUNNING: The process which is using the CPU
* READY: The process which is in ready Queue and waiting to Use CPU
* WAITING: The process is waiting for some I/O.
* DONE: The process had completed its execution.

The Program process- run.py can takes several arguments via command line before starting execution…

They are

* **l: PROCESS\_LIST**
* a comma-separated list of processes to run, in the

form X1:Y1, X2:Y2... where X is the number of

instructions that process should run, and Y the

chances (from 0 to 100) that an instruction will use

the CPU or issue an IO

* **L: IO\_LENGTH --iolength=IO\_LENGTH**
* How long can I/O take?
* **-S PROCESS\_SWITCH\_BEHAVIOR, --switch=PROCESS\_SWITCH\_BEHAVIOR**
* when to switch between processes: SWITCH\_ON\_IO,

SWITCH\_ON\_END

* **-I IO\_DONE\_BEHAVIOR, --iodone=IO\_DONE\_BEHAVIOR**
* type of behaviour when IO ends: IO\_RUN\_LATER,

IO\_RUN\_IMMEDIATE

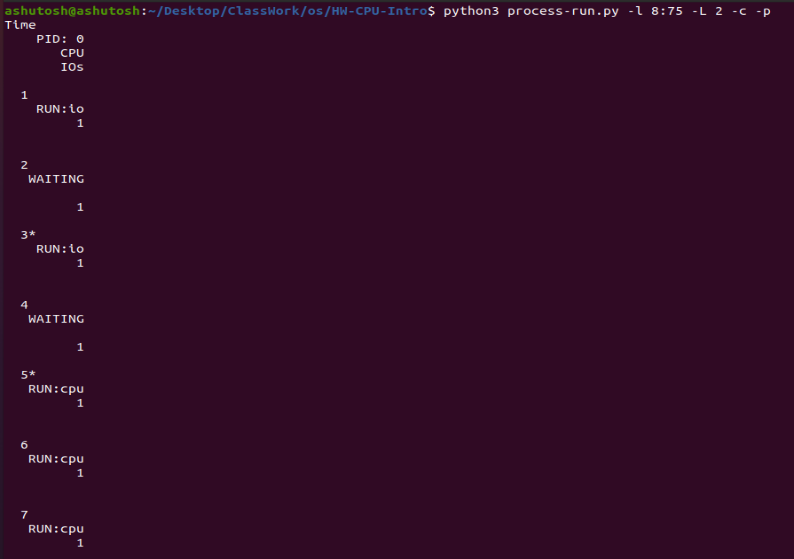
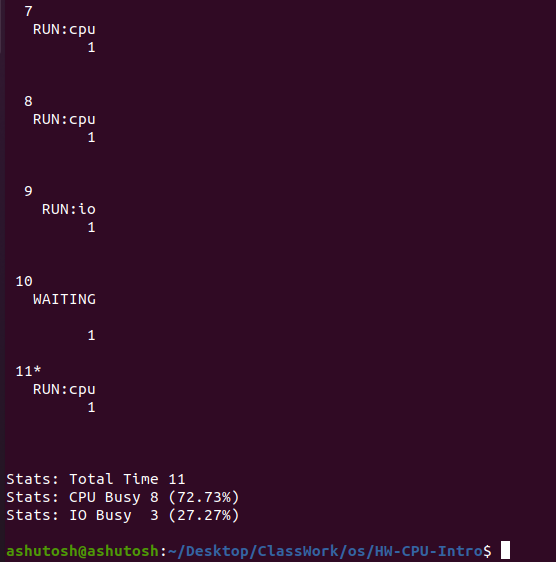
* **-c**
* compute answers for me
* **-p, --print stats**
* print statistics at end; only useful with -c flag

(otherwise, stats are not printed)

Let’s run a process having 8 instructions ,75 out of 100 chances that it can use CPU or issue an I/O and IO length as 2 and computes its answer and stats for this we have to use command:

**python3 process-run.py -l 8:75 -L 2 -c -p**

The output I got is:

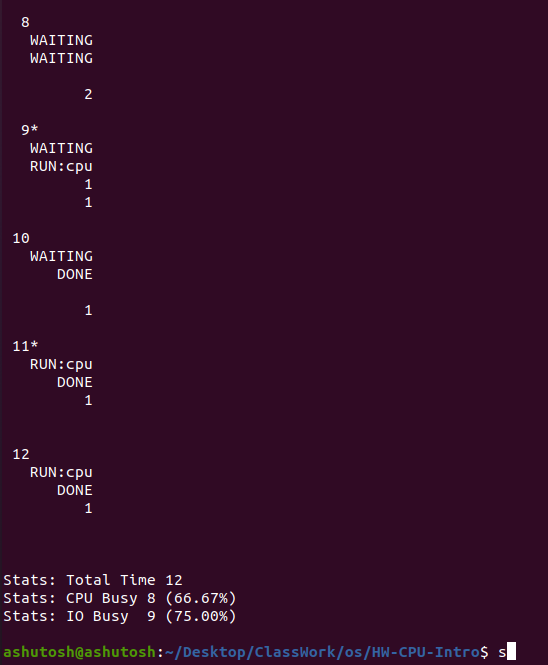
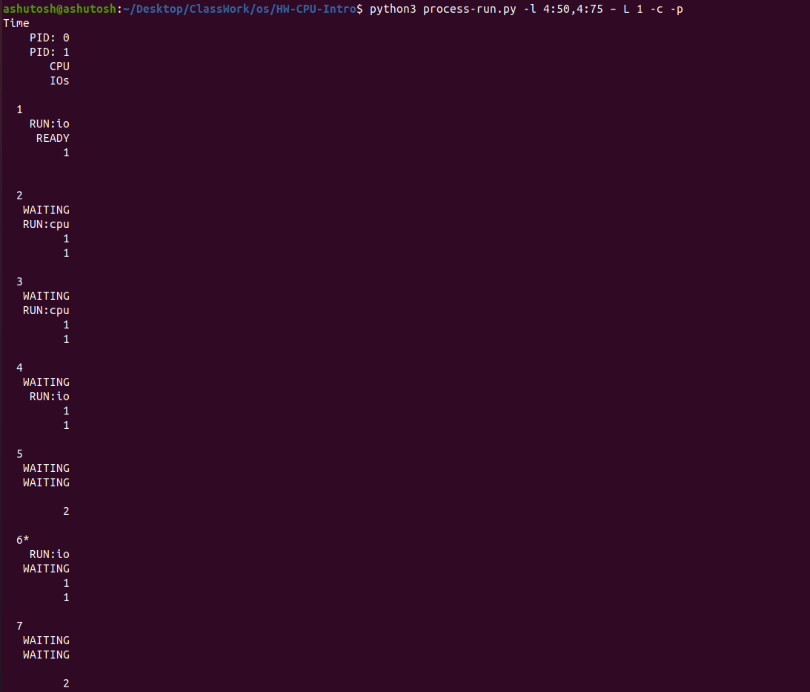


In the above the first line shows clock tick and then the process state and then number if CPU is used otherwise blank and then in next line it shows number if I/O is used otherwise black and have 1 line break.

Now Let’s run 2 process simultaneously having 4 instructions each and one have 50 by 100 chances and other have 75 by 100 chances that it can use CPU or issue an I/O and have IO length of 1 for each process. Let’s compute its answer and stats, for this our command be like…

**python3 process-run.py -l 4:50,4:75 – L 1 -c -p**

The output I got is:



1. Process API

Answer: In this there are system calls like fork (), wait (), exec () is programmed and we have to analyse accordingly the process that are running and some other experiments like passing variable and getting that are they passed successfully or not in this there are some programs written in c and we have executed them all accordingly to given guidelines this is written in C language.

1. Scheduling Basics (scheduler.py)

Answer: This program gives us platform to analyse different process under different scheduling metrics such as response time, turnaround time, and wait time per process as well as the average response time, average turnaround time and average wait time for all the given process.

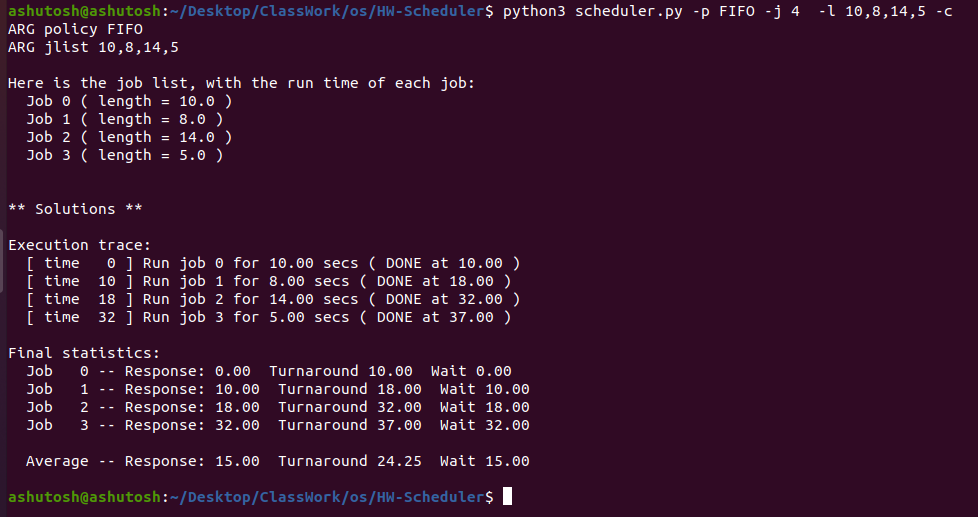
We can send via arguments some arguments such the type of scheduling, number of process or job, random seed number, the length of the process and the maximum length of the process…

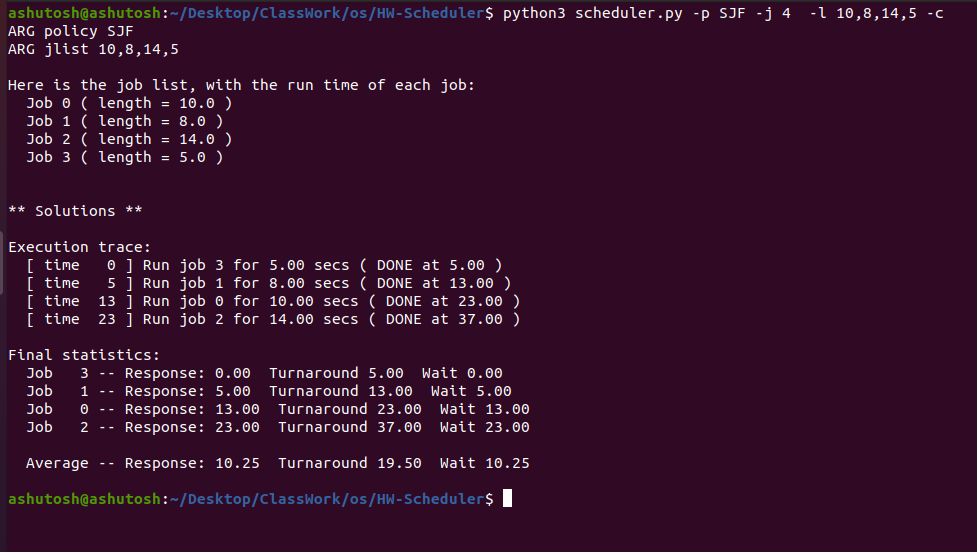
**-p <type of scheduling> -j <number of jobs> -l <job-length of each job> -c**

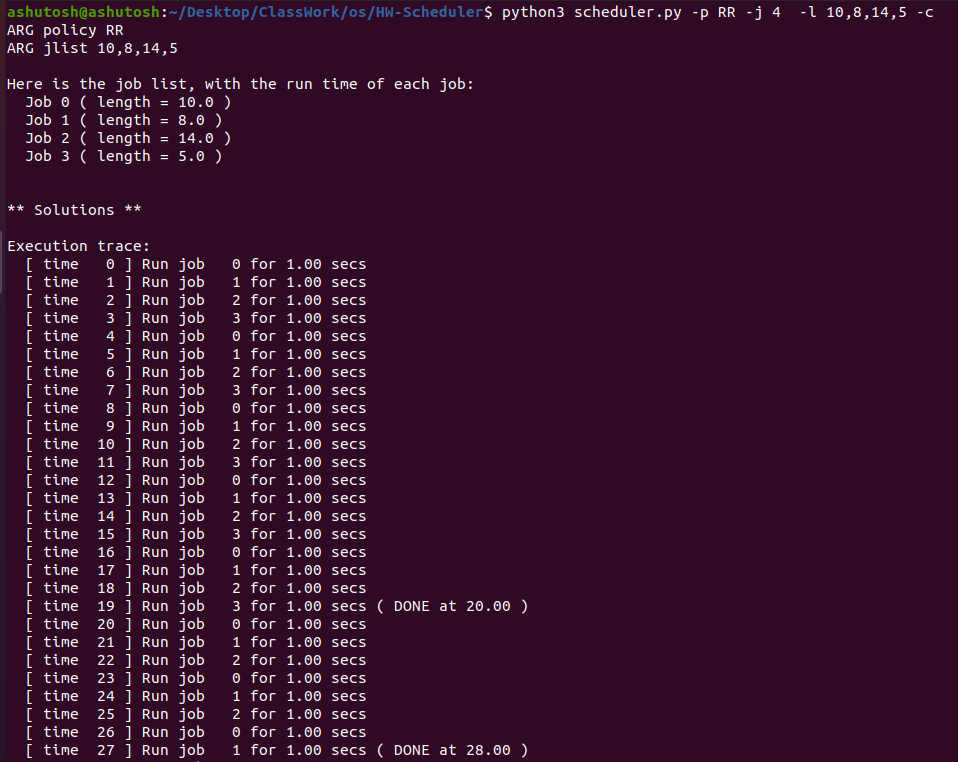
c is used at last for displaying the information of job...

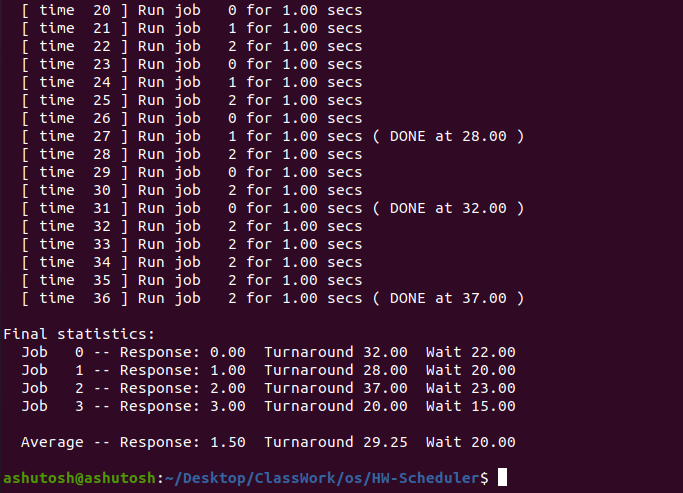
Let’s us suppose we have 4 job of length 10,8,14,5 …...

* Using First Come First Serve FCFS (FIFO) scheduling policy



* Using Short Job First SJF scheduling policy
* Using Round Robin RR scheduling policy





1. MLFQ Scheduling (mlfq.py)

Answer: This program is an implementation of Multi Level Feedback Queue scheduling. This program chooses the job according to their priority and then lead to job done. In this type of scheduling approach there are multiple level of queue s and used feedback to determine the priority of a given job. History is its guide: pay attention to how job behave over time and treat them accordingly.

There are some rules related with it:

* Rule 1: If Priority(A) > Priority(B), A runs (B doesn’t).
* Rule 2: If Priority(A) = Priority(B), A & B run in round-robin fashion using the time slice (quantum length) of the given queue.
* Rule 3: When a job enters the system, it is placed at the highest priority (the topmost queue).
* Rule 4: Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (i.e., it moves down one queue).
* Rule 5: After some time

Some important terms which are used to give argument via command for this program are ….

* -s SEED, --seed = SEED the random seed
* n NUMQUEUES, --numQueues=NUMQUEUES number of queues in MLFQ
* q QUANTUM --quantum= QUANTUM length of time slice
* Q QUANTUMLIST, --quantumList=QUANTUMLIST length of time slice per queue level,

Specified as x, y, z, … where x is the quantum length for the highest- priority

Queue, y the next highest and so forth.

* -j NUMJOBS, --numJobs=NUMJOBS

number of jobs in the system

* -m MAXLEN, --maxlen=MAXLEN

max run-time of a job (if random)

* -M MAXIO, --maxio=MAXIO

max I/O frequency of a job (if random)

* -B BOOST, --boost=BOOST

how often to boost the priority of all?

jobs back to high priority (0 means never)

* -i IOTIME, --iotime=IOTIME

how long can I/O should last (fixed constant)

* -S, --stay reset and stay at same priority level

when issuing I/O

* -l JLIST, --jlist=JLIST

a comma-separated list of jobs to run,

in the form x1, y1, z1:x2, y2, z2... where

x is start time, y is run time, and z

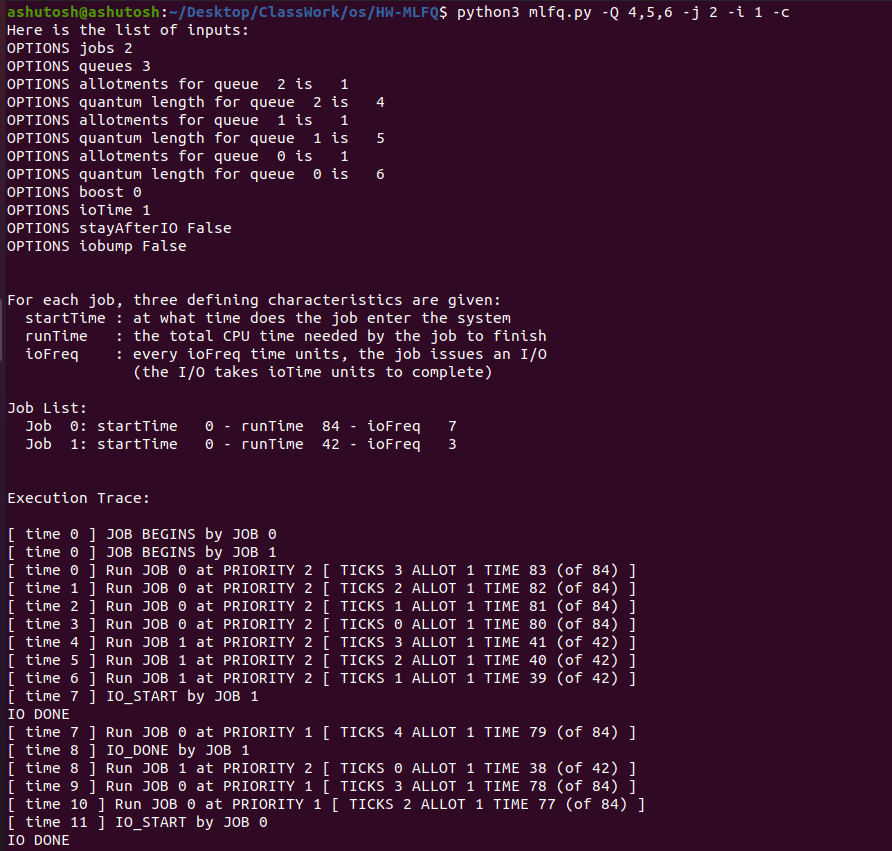
is how often the job issues and I/O request

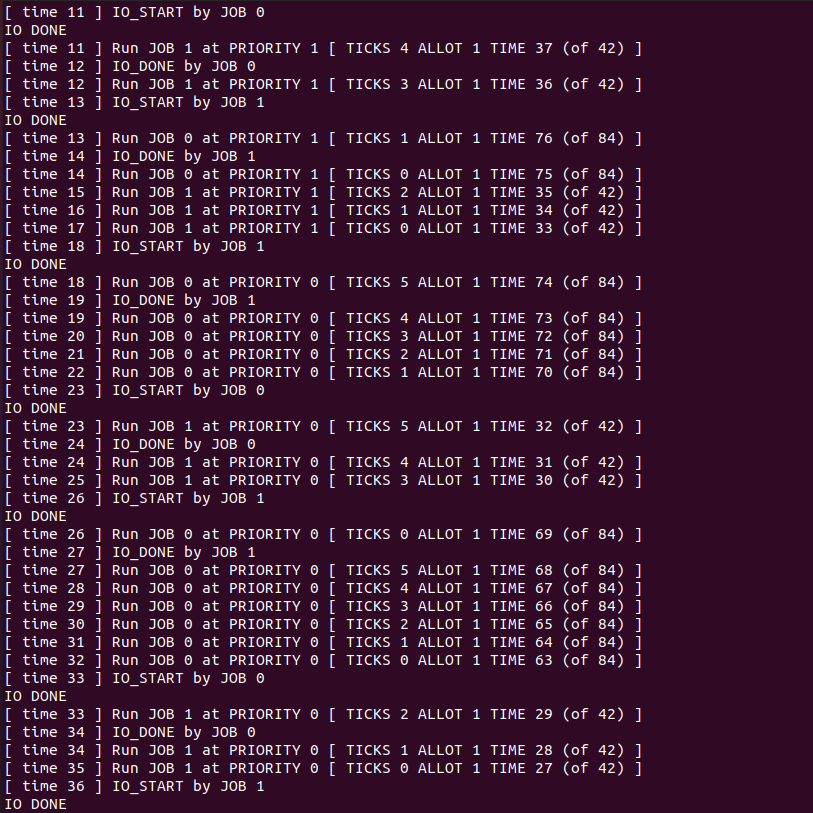
* -c compute answers for me

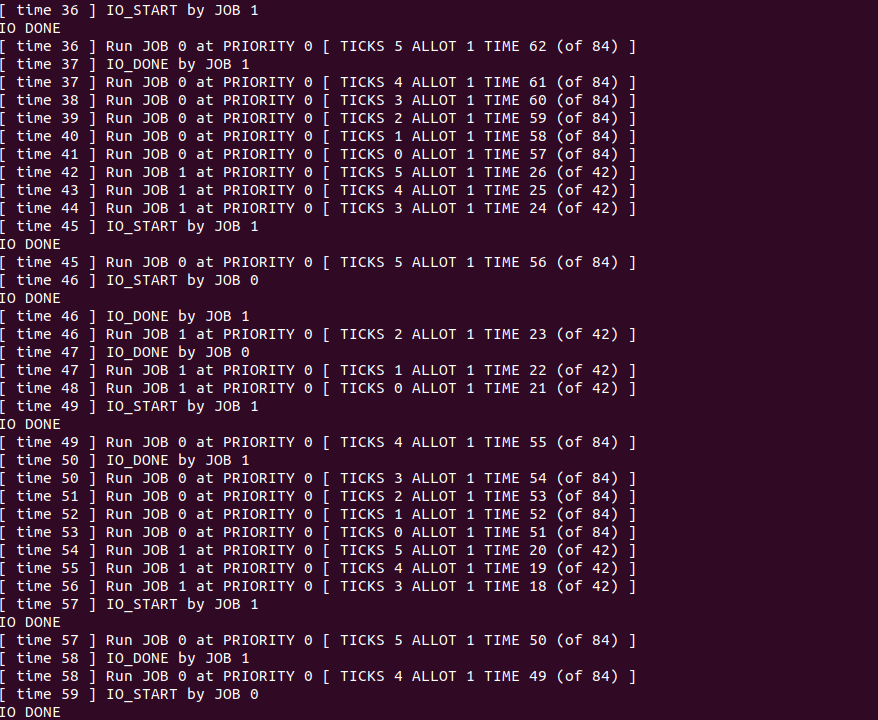
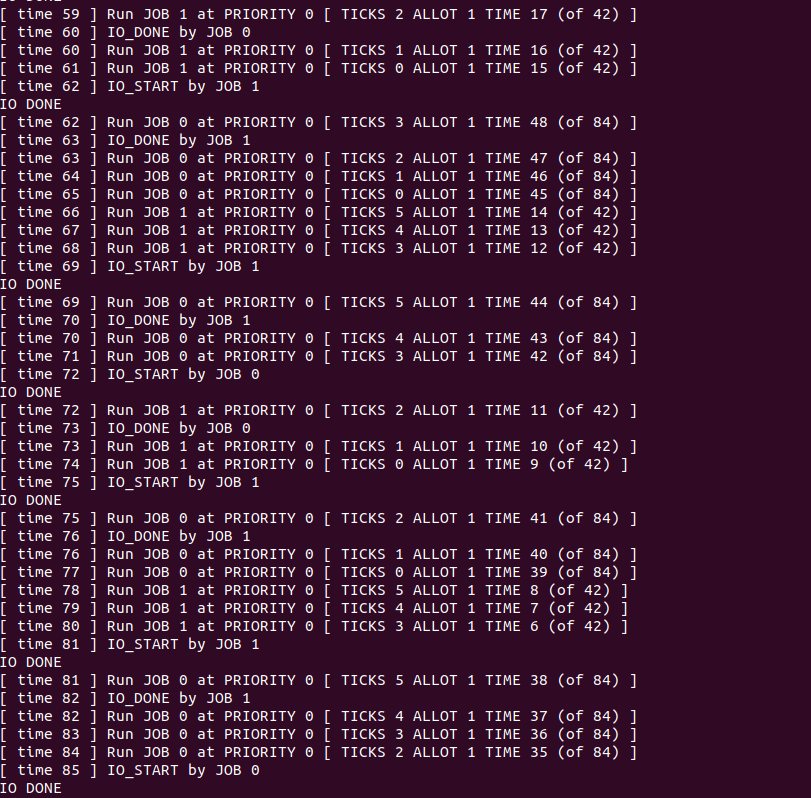
Lets the quantum length of queue is 4, 5, 6 respectively and I/O time is 1 and c for describing how it runs. The command for that will be...

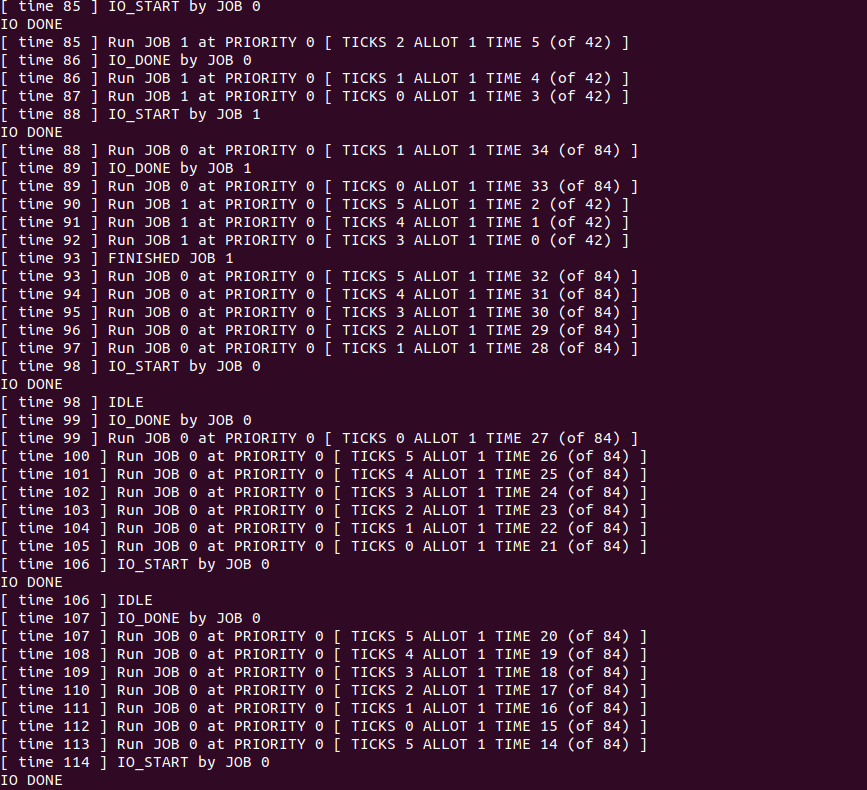
python3 mlfq.py -Q 4,5,6 -j 2 -i 1 -c

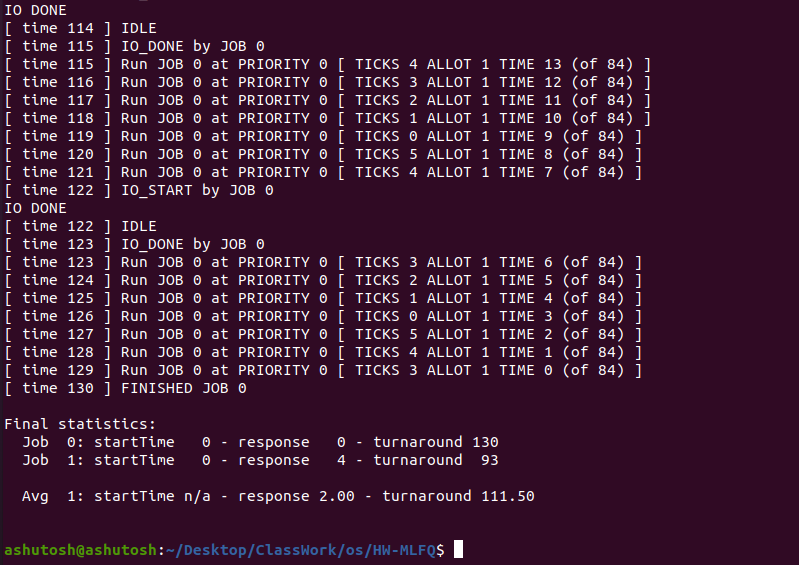
The output is ….











5) Lottery Scheduling (lottery.py)

This program is about lottery scheduling of the process. This program help us to analyse how the lottery scheduling takes place lead to job done. Some commands which we can use in order to analyse how the CPU and IO participates are:

* -h, --help

show this help message and exit

* -s SEED, --seed=SEED

the random seed

* -j JOBS, --jobs=JOBS

number of jobs in the system

* -l JLIST, --jlist=JLIST

instead of random jobs, provide a comma-separated list

of run times and ticket values (e.g., 10:100,20:100

would have two jobs with run-times of 10 and 20, each

with 100 tickets)

* -m MAXLEN, --maxlen=MAXLEN

max length of job

* -T MAXTICKET, --maxtick=MAXTICKET

maximum ticket value, if randomly assigned

* -q QUANTUM, --quantum=QUANTUM

length of time slice

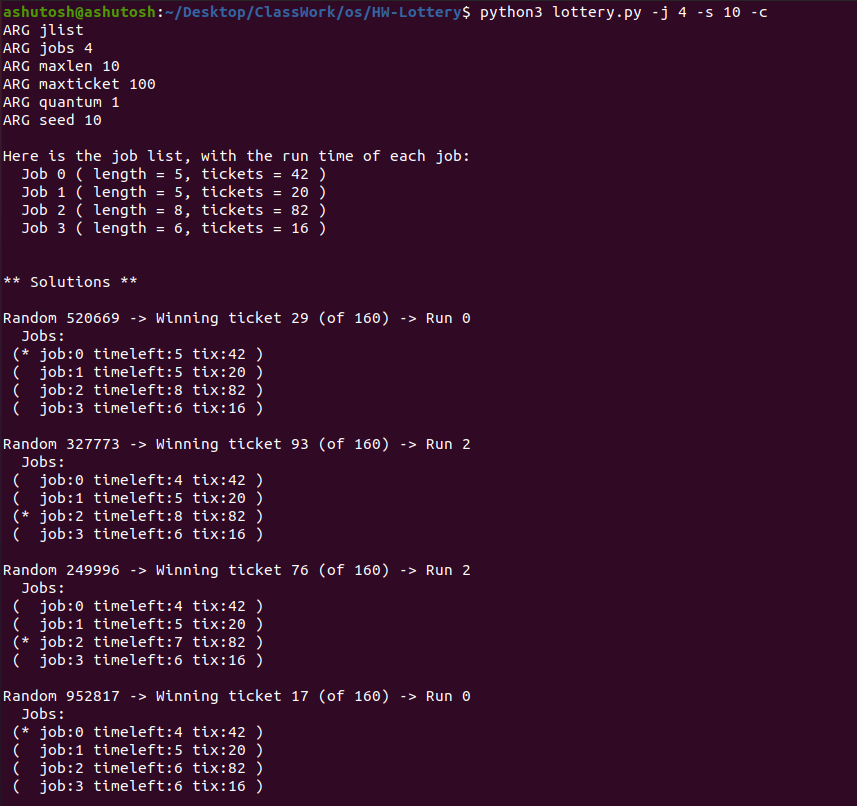
* -c, --compute

compute answers for me

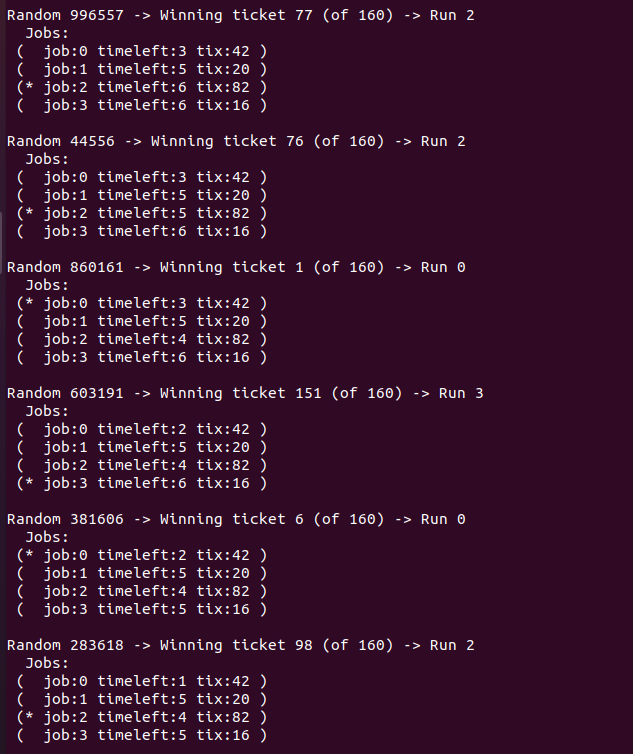
Let’s suppose we have 4 jobs and we have to analyse this ….

For that the command I use is

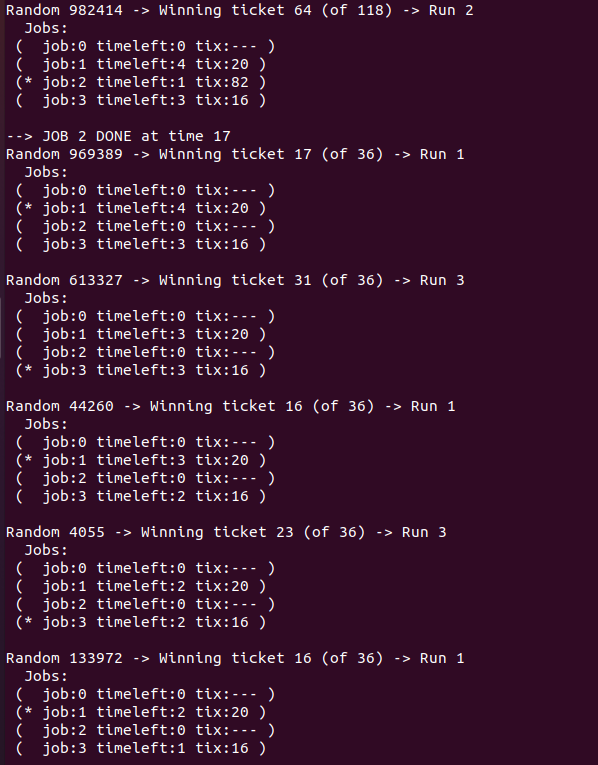
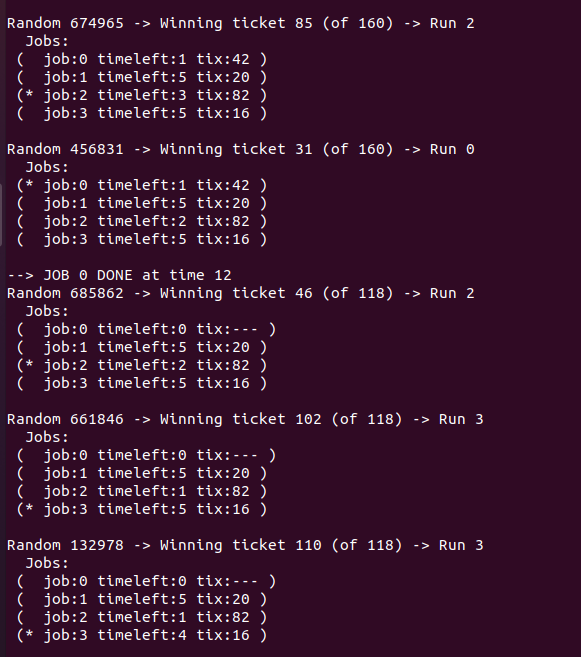
**python3 lottery.py -j 4 -s 10 -c**



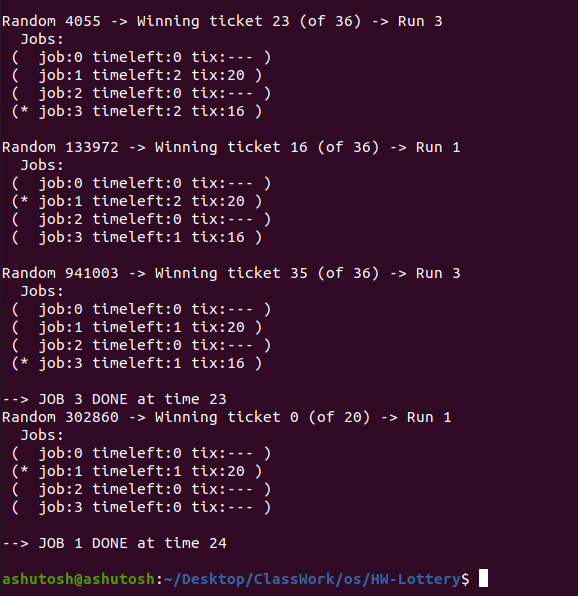
1.

2.

3.

4.

5.



There are so many functions that I learned Some of them are...

1. move\_to\_ready
2. move\_to\_wait
3. move\_to\_running
4. move\_to\_done

As the name suggests these function move the job to ready state ,wait state, running state and done state accordingly..