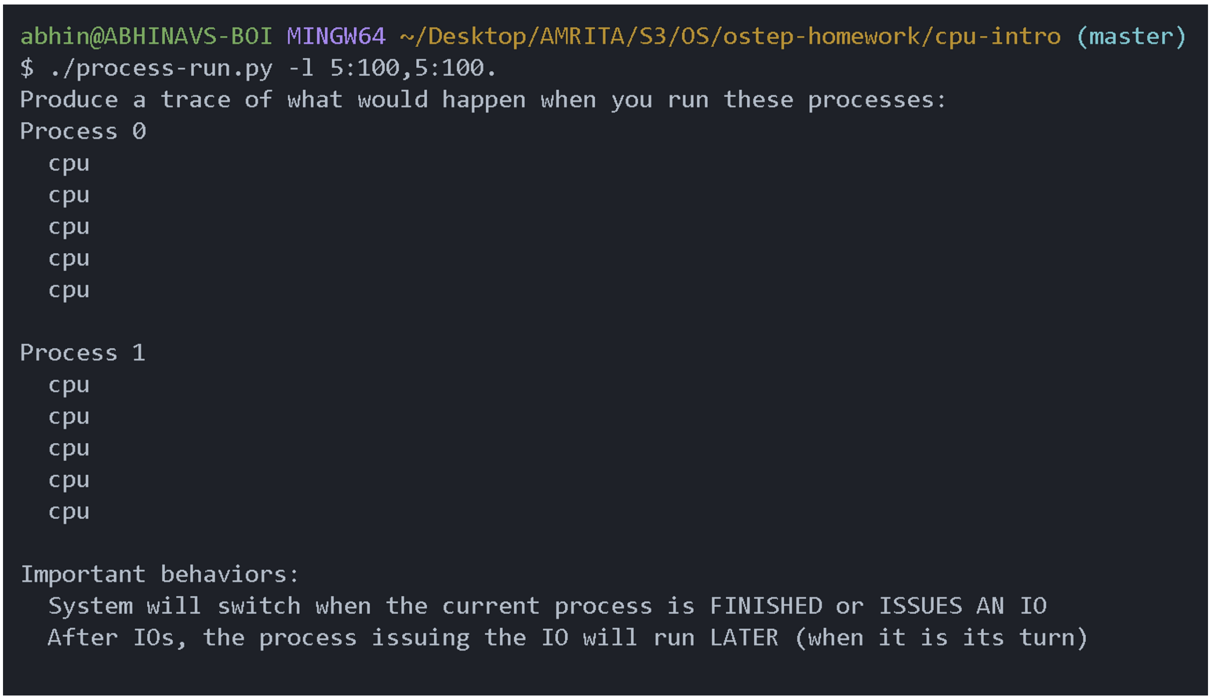
Name: Abhinav Pandey Roll No: AM.EN.U4AIE21088

ASSIGNMENT 1

QUESTION 1:

Run process-run.py with the following flags: -l 5:100,5:100. What should the CPU utilisation be (e.g., the percent of time the CPU is in use?) Why do you know this?

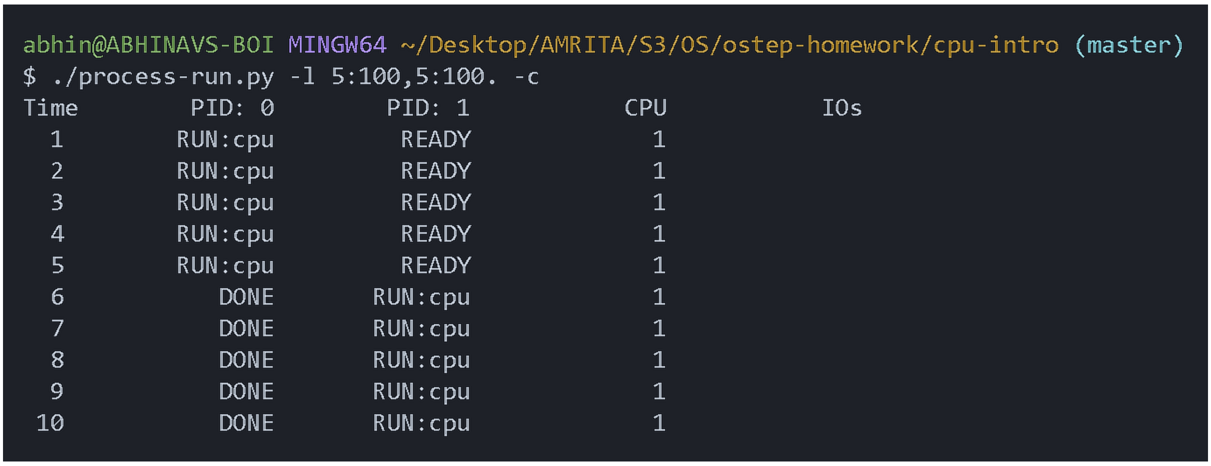


```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 5:100,5:100.
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu
  cpu

Process 1
  cpu
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch when the current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO will run LATER (when it is its turn)
```

Here we are running 5 instructions and 100 percent of the cpu is being utilised.

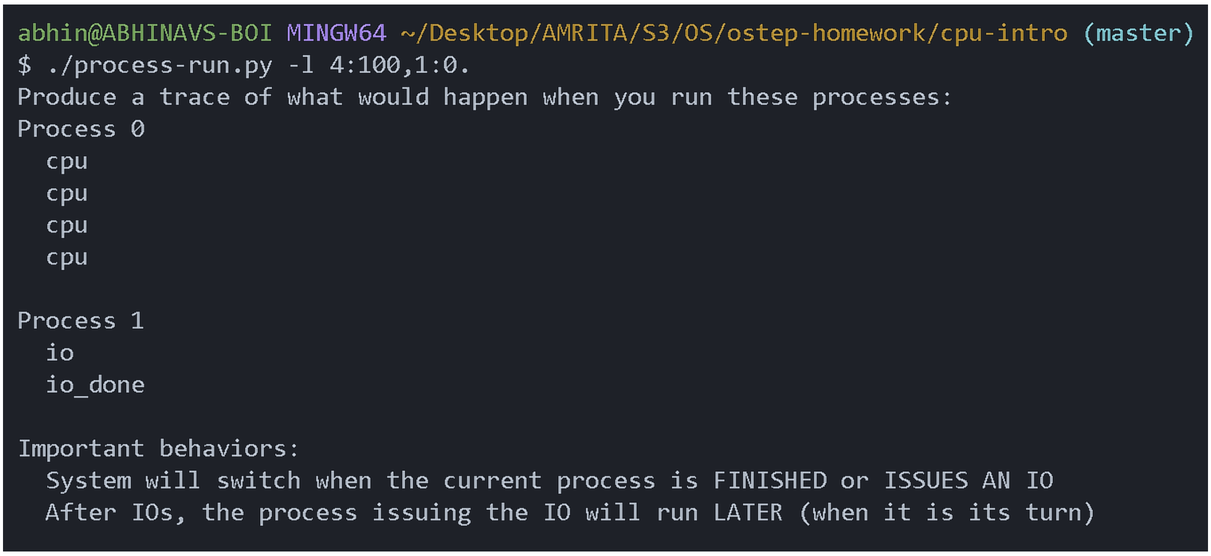


```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 5:100,5:100. -c
Time           PID: 0         PID: 1           CPU           IOs
  1         RUN:cpu          READY             1
  2         RUN:cpu          READY             1
  3         RUN:cpu          READY             1
  4         RUN:cpu          READY             1
  5         RUN:cpu          READY             1
  6            DONE        RUN:cpu             1
  7            DONE        RUN:cpu             1
  8            DONE        RUN:cpu             1
  9            DONE        RUN:cpu             1
 10            DONE        RUN:cpu             1
```

We know this because the system will use 100% of its capability and switch tasks after one is completed.

QUESTION 2:

Now run with these flags: ./process-run.py -l 4:100,1:0. These flags specify one process with 4 instructions (all to use the CPU), and one that simply issues an I/O and waits for it to be done. How long does it take to complete both processes?

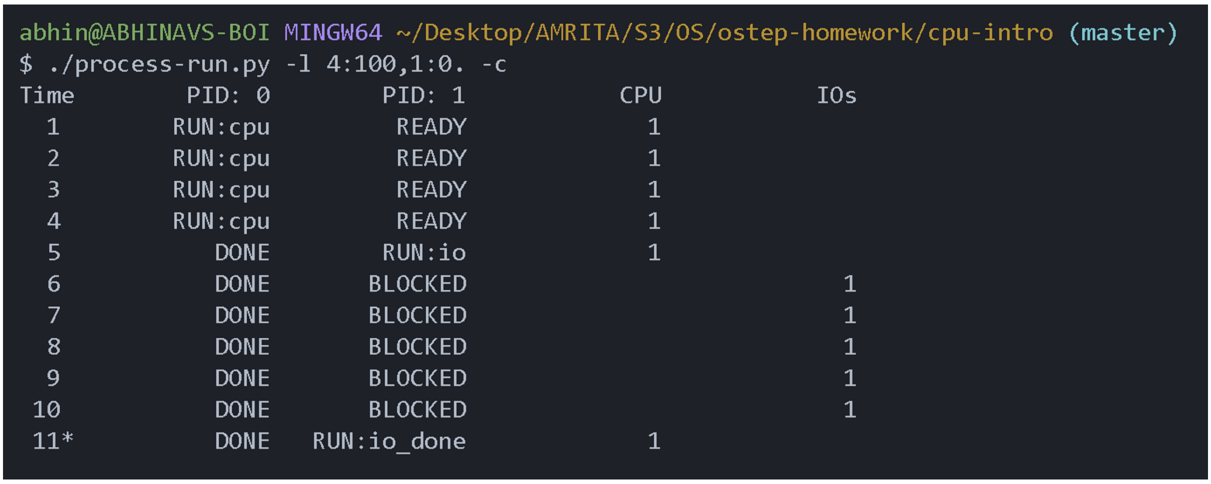


```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 4:100,1:0.
Produce a trace of what would happen when you run these processes:
Process 0
  cpu
  cpu
  cpu
  cpu

Process 1
  io
  io_done

Important behaviors:
  System will switch when the current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO will run LATER (when it is its turn)
```

To complete both the processes it will take 11 ticks. The first one is a CPU process with 4 lines, so it will take 4 ticks. The next process is an I/O process. Since the I/O length is not specified, by default it is set to 5. So first the CPU runs for 1 tick and then the I/O runs for 5 ticks (default value) which makes it 6 ticks. Then the simulator adds an empty tick to indicate the completion of the I/O process.

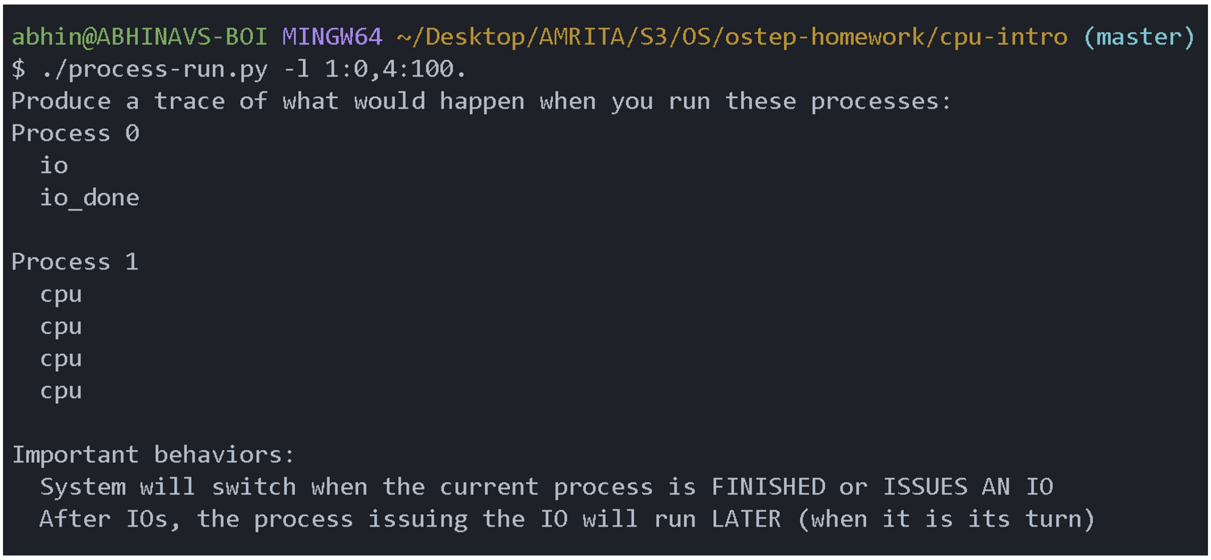


```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 4:100,1:0. -c
Time        PID: 0       PID: 1          CPU          IOs
  1        RUN:cpu        READY            1
  2        RUN:cpu        READY            1
  3        RUN:cpu        READY            1
  4        RUN:cpu        READY            1
  5           DONE       RUN:io            1
  6           DONE      BLOCKED                           1
  7           DONE      BLOCKED                           1
  8           DONE      BLOCKED                           1
  9           DONE      BLOCKED                           1
 10           DONE      BLOCKED                           1
 11*          DONE   RUN:io_done           1
```

QUESTION 3:

Switch the order of the processes: -l 1:0,4:100. What happens now?

Does switching the order matter? Why?

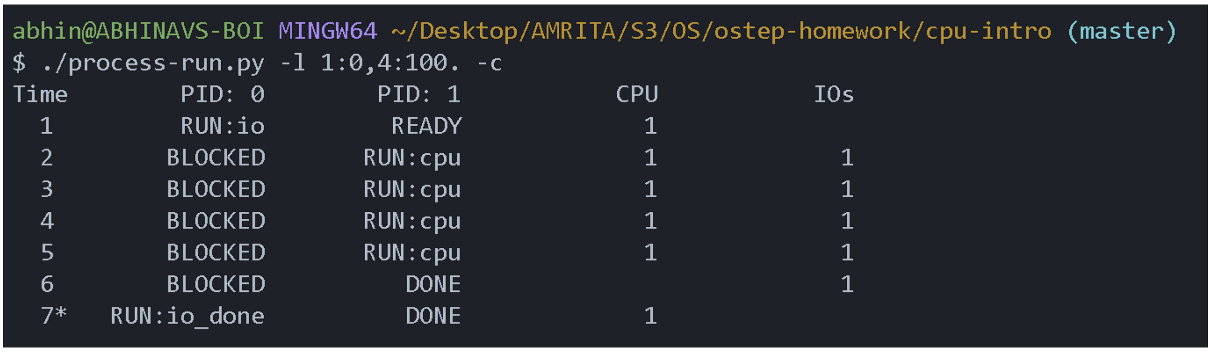


```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 1:0,4:100.
Produce a trace of what would happen when you run these processes:
Process 0
  io
  io_done

Process 1
  cpu
  cpu
  cpu
  cpu

Important behaviors:
  System will switch when the current process is FINISHED or ISSUES AN IO
  After IOs, the process issuing the IO will run LATER (when it is its turn)
```

Here, it takes a total of 7 ticks to complete the processes. First the IO process runs and it takes 5 ticks as the CPU remains idle. So, during this time, the CPU runs the next process of 4 lines.



```
abhin@ABHINAVS-BOI MINGW64 ~/Desktop/AMRITA/S3/OS/ostep-homework/cpu-intro (master)
$ ./process-run.py -l 1:0,4:100. -c
Time        PID: 0        PID: 1           CPU           IOs
  1         RUN:io         READY             1
  2        BLOCKED       RUN:cpu             1             1
  3        BLOCKED       RUN:cpu             1             1
  4        BLOCKED       RUN:cpu             1             1
  5        BLOCKED       RUN:cpu             1             1
  6        BLOCKED          DONE                           1
  7*   RUN:io_done          DONE             1
```