1. Heap and stack are copied but not shared, only shared memory segments are shared.
2. 15. Value is on the stack and the stack is copied at fork(), so the child process can't change the value variable as they exist in different processes.
3. 2^3 = 8. At each line the process branches into 2, which it does 3 times.
4. Task parallelism.
5. Kernel threads are visible to the os and can achieve true parallelism. Parallelism of user level threads depends on the mapping between user threads and kernel threads that is implemented in a runtime. To create and manage kernel threads syscalls are needed which have more overhead than user threads.
6. Interrupt or System Call
   Save State
   Update Process Control Block
   Switch Memory Context
   Choose Next Thread
   Load State
   Update Scheduler
   Return to User Mode
   Resume Execution

7. A preemptive process runs for a set amount of time and is then put back in the ready-queue with other processes. The amount of time can either be a simple timer/value or if another process with higher priority is in the ready queue.
   A non-preemptive process runs until either it needs an I/O burst or the process terminates itself.
8. a) ((8-0.0) + (12-0.4) + (13-1.0)) / 3 = 31.6/3 = 10.5333… ms
   b) ((8-0.0) + (9-1.0) + (13-0.4)) / 3 = 28.6/ 3 = 9.5333… ms
   c) ((1+0) + (1.6 + 4) + (6 + 8))/3 = 20.6/3 = 6.8666….7 ms
9. a. Insert pics
   b. FCFS:
      SJF:
      Non-Preemptive:
      RR: P1=2, P2=3,P3=20, P4=13, P5=18

1

Heap and stack are copied but not shared, only shared memory segments are shared.

2

15. Value is on the stack and the stack is copied at fork(), so the child process can’t change the value variable as they exist in different processes.

3

2^3 = 8. At each line the process branches into 2, which it does 3 times.

4

Task parallelism.

5

Kernel threads are visible to the os and can achieve true parallelism. Parallelism of user level threads depends on the mapping between user threads and kernel threads that is implemented in a runtime. To create and manage kernel threads syscalls are needed which have more overhead than user threads.

6

Interrupt or System Call
Save State
Update Process Control Block
Switch Memory Context
Choose Next Thread
Load State
Update Scheduler
Return to User Mode
Resume Execution

7

A preemptive process runs for a set amount of time and is then put back in the ready-queue with other processes. The amount of time can either be a simple timer/value or if another process with higher priority is in the ready queue.

A non-preemptive process runs until either it needs an I/O burst or the process terminates itself.

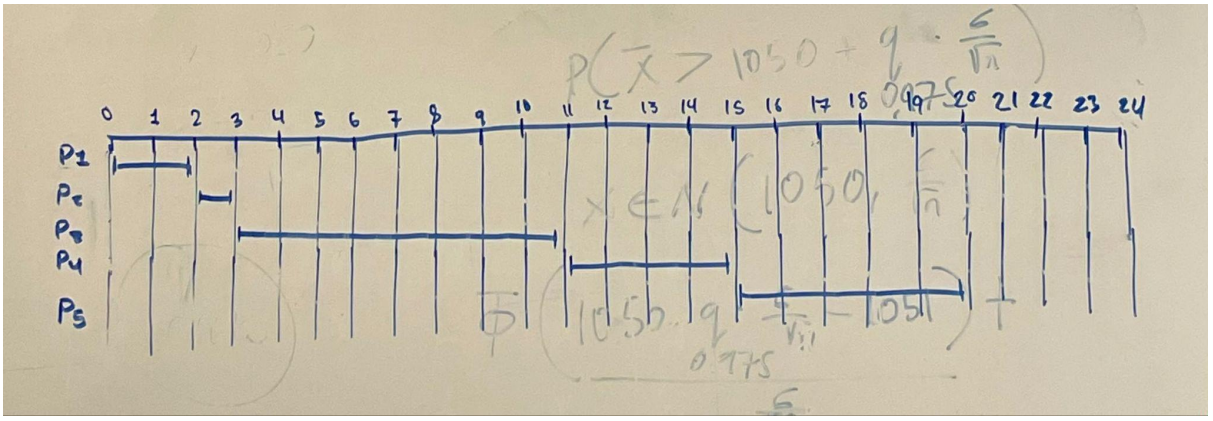
8

a) ((8-0.0) + (12-0.4) + (13-1.0)) / 3 = 31.6/3 = 10.5333… ms
b) ((8-0.0) + (9-1.0) + (13-0.4)) / 3 = 28.6/ 3 = 9.5333… ms
c) ((1+0) + (1.6 + 4) + (6 + 8)) / 3 = 20.6/3 = 6.8666….7 ms

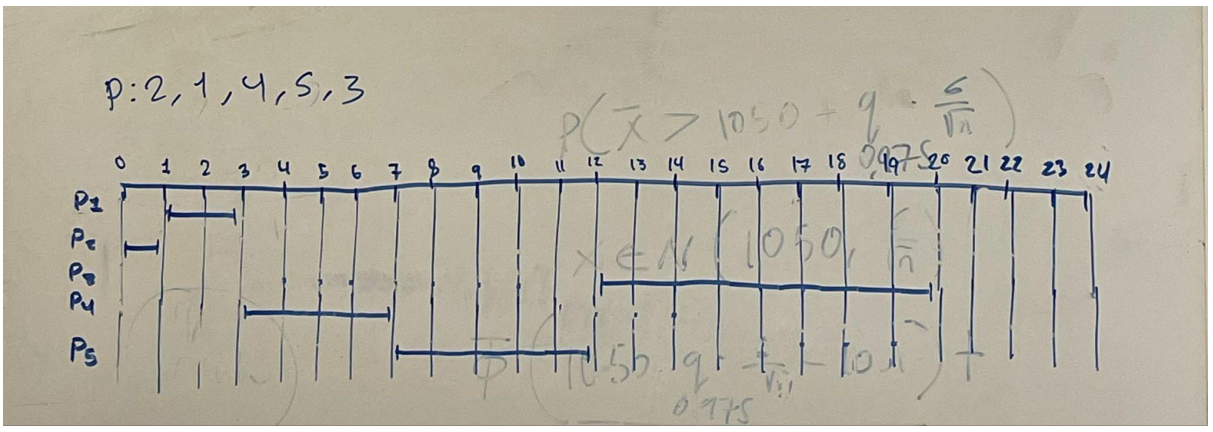
9

a)

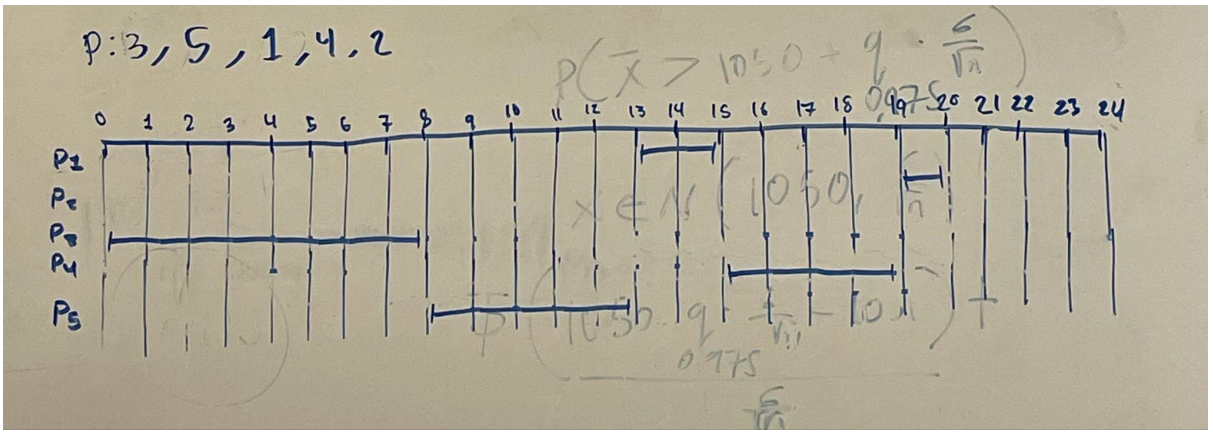
FCFS:



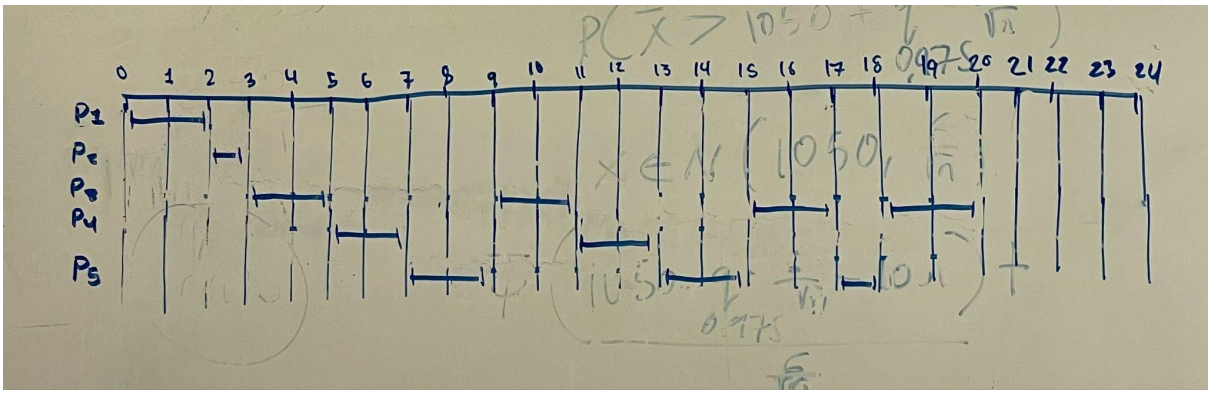
SJF:



Non-preemptive priority:



RR:



b)

FCFS: P1=2, P2=3, P3=11, P4=15, P5=20

SJF: P1=3, P2=1, P3=23, P4=7, P5=12

Non-Preemptive priority: P1=15, P2=20, P3=8, P4=19, P5=13

RR: P1=2, P2=3,P3=20, P4=13, P5=18

c)

FCFS: P1=0, P2=2, P3=3, P4=11, P5=15
SJF: P1=1, P2=0, P3=12, P4=3, P5=7
Non-Preemptive priority: P1=13, P2=19, P3=0, P4=15, P5=8
RR: P1=0, P2=2, P3=12, P4=9, P5=13

d)

FCFS: 31/3
SJF: 23/3
Non-Preemptive priority: 55/3
RR: 36/3

Answer: SJF