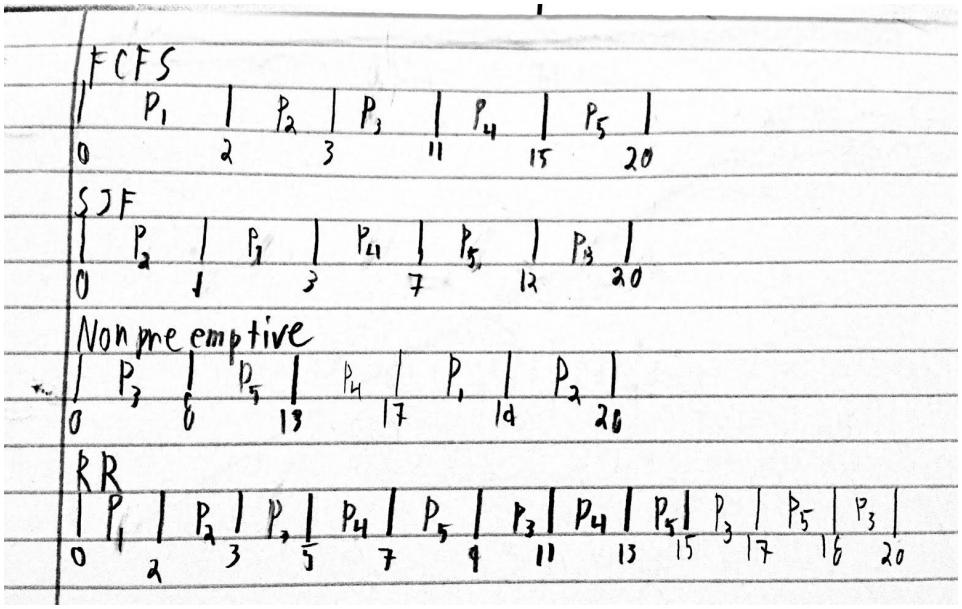
1: Both programs produce 4 processes each. Putting the print statement outside the loop just alters the number of line printed, not the number of processes created.

3.26: I attached the C code in a zip file.

6.16:a.



b.

FCFS

P1: 2+0=2

P2: 2+1=3

P3: 8+3=11

P4: 4+11=15

P5: 5+15=20

SJF

P1: 2+1=3

P2: 1+0=1

P3: 8+12=20

P4: 4+3=7

P5: 5+7=12

Nonpreemptive

P1: 2+17=19

P2: 1+19=20

P3: 8+0=8

P4: 4+13=17

P5: 5+8=13

RR
P1: 2+0=2
P2: 1+2=3
P3: 8+3+4+4+1=20
P4: 4+5+4=13
P5: 5+7+4+2=18
c.
FCFS
P1: 0
P2: 2
P3: 3
P4: 11
P5: 15
SJF
P1: 1
P2: 0
P3: 12
P4: 3
P5: 7
Nonpreemptive
P1: 17
P2: 19
P3: 0
P4: 13
P5: 8
RR
P1: 0
P2: 2
P3: 3+4+4+1=12
P4: 5+4=9
P5: 7+4+2=13
d. SJF with an average of seconds 4.6
6.19 Priority and SJF scheduling may suffer from starvation because in a heavily loaded CPU, both may prevent too low of priority or too long of a job from ever getting CPU time.
6.22 A strategy to maximize the CPU time is to break your process into smaller processes so that each process will have a higher priority for SJF and more opportunities to run in Round Robin.
6.24 a. FCFS discriminates against shorter jobs because short jobs that arrive after long jobs are forced to wait for the longs jobs to finish.
b. RR gives all jobs, regardless of length, an equal amount of CPU time to execute, which will allow short jobs to complete sooner since they can finish in less time quantums.