Q1.

CPU-bound processes tend to have longer CPU burst times, if an algorithm can handle the processes with long CPU burst time quicker, we say it favors CPU bound processes.
In this way, FCFS may be the best one since long CPU burst processes can not be preempted.
Simulation results:
Algorithm SJF
– average CPU burst time: 3399.893 ms
– average wait time: 23552.856 ms
– average turnaround time: 26956.749 ms
– total number of context switches: 466
– total number of preemptions: 0
Algorithm SRT
– average CPU burst time: 3399.893 ms
– average wait time: 23620.721 ms
– average turnaround time: 27027.515 ms
– total number of context switches: 804
– total number of preemptions: 338
Algorithm FCFS
– average CPU burst time: 3399.893 ms
– average wait time: 20919.760 ms
– average turnaround time: 24323.652 ms
– total number of context switches: 466
– total number of preemptions: 0
Algorithm RR
– average CPU burst time: 3399.893 ms
– average wait time: 22932.912 ms
– average turnaround time: 26445.758 ms
– total number of context switches: 13159
– total number of preemptions: 12693

The above simulation results approve that for CPU-bound processes, FCFS algorithm which has a shortest average wait time would be the best-suited one.

I/O bound processes tend to have shorter CPU burst times and are likely to spend most of their time in I/O. So for I/O-bound processes, SRT algorithm would be the better one. Since it can make the processes with shorter CPU burst time to use CPU and then get I/O block quicker. This would let more processes to stay at I/O block and then increase the I/O blocking efficiency.
Simulation results:
Algorithm SJF
– average CPU burst time: 67.998 ms
– average wait time: 19.358 ms
– average turnaround time: 91.356 ms
– total number of context switches: 466
– total number of preemptions: 0
Algorithm SRT
– average CPU burst time: 67.998 ms
– average wait time: 15.358 ms
– average turnaround time: 87.433 ms
– total number of context switches: 475
– total number of preemptions: 9
Algorithm FCFS
– average CPU burst time: 67.998 ms
– average wait time: 17.264 ms
– average turnaround time: 89.262 ms
– total number of context switches: 466

– total number of preemptions: 0
Algorithm RR
– average CPU burst time: 67.998 ms
– average wait time: 19.292 ms
– average turnaround time: 91.453 ms
– total number of context switches: 485
– total number of preemptions: 19

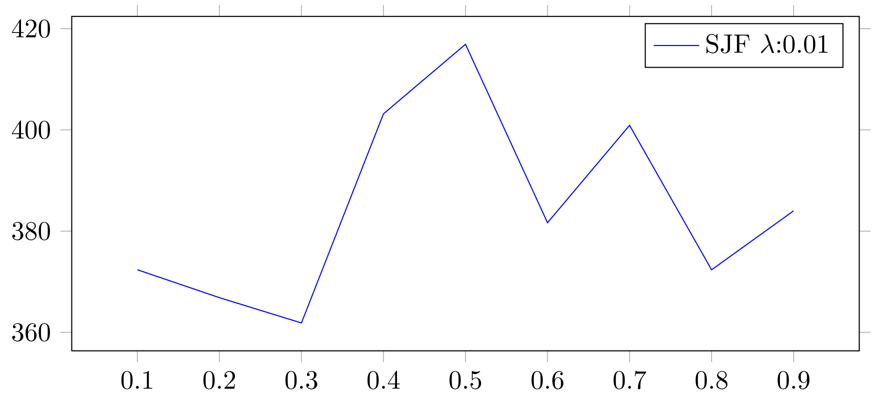
The above simulation results approve that for IO-bound processes, SRT algorithm which has a shortest average wait time would be the best-suited one.

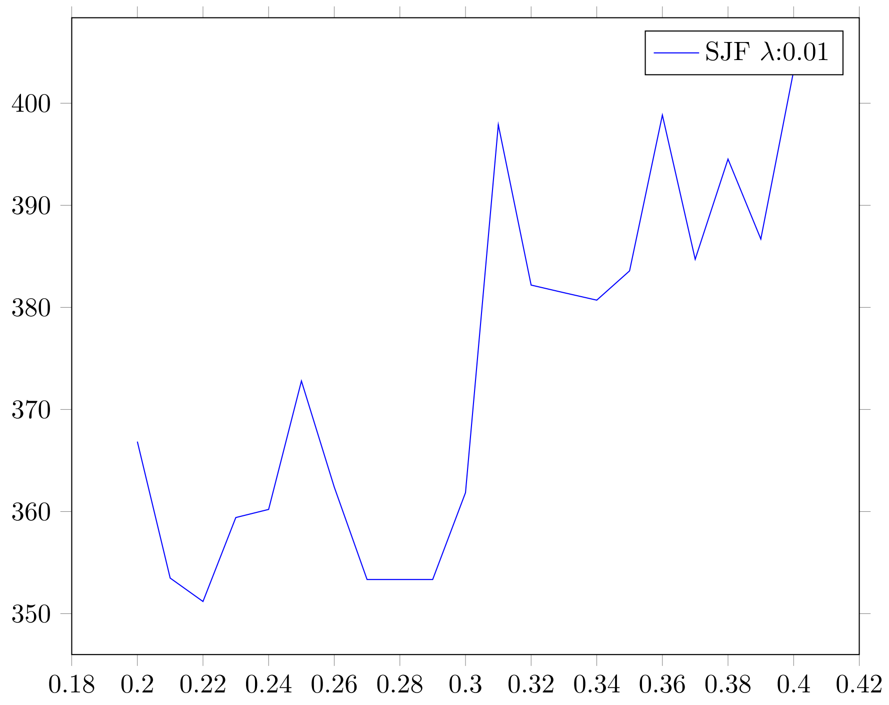
Q2.
If we change the *rr_add* from END to BEGINNING:
1) this "breaks" the FAIRNESS ides.
2) the advantage of this approach is that for I/O-bound processes, they would get through their CPU bursts quickly and get back to doing more I/O. So this approach works better for I/O-bound processes.
Simulation results:
1). *rr_add* = END
Algorithm RR
– average CPU burst time: 67.998 ms
– average wait time: 19.292 ms
– average turnaround time: 91.453 ms
– total number of context switches: 485
– total number of preemptions: 19

2). *rr_add* = BEGINNING
Algorithm RR
– average CPU burst time: 67.998 ms
– average wait time: 16.105 ms
– average turnaround time: 88.283 ms
– total number of context switches: 487
– total number of preemptions: 21

The above simulation results approve that changing adding order from end to beginning of the queue, for I/O-bound processes, this would make a better CPU scheduling result.

Q3.
In order to seek the value of α producing the best results, we try to plot a graph with x axis is α value and y axis is the corresponding wait time from the simulation results. We can get the best α value corresponding to the smallest wait time value from the curve.





In the first graph, we find the best α is around 0.3. In order to get more precise result, we further plot the second graph with α in range [0,2, 0.4]. We get the best α value is around 0.22.
So in this case with $\lambda = 0.01$, $\alpha = 0.22$ seemed to produce the best results.

Q4.
Using Shortest remaining time algorithm, with adding preemption, can handle short processes quicker than using shortest job first algorithm.
It means that adding preemption can be better at getting I/O-bound processes through the CPU more quickly.
Simulation results:
Algorithm SJF
– average CPU burst time: 67.998 ms
– average wait time: 19.358 ms
– average turnaround time: 91.356 ms
– total number of context switches: 466
– total number of preemptions: 0
Algorithm SRT
– average CPU burst time: 67.998 ms
– average wait time: 15.358 ms
– average turnaround time: 87.433 ms
– total number of context switches: 475
– total number of preemptions: 9

Q5.
1. In this simulation, we only consider single processor scheduling. In a real-world operating system, it would have multiple-processor scheduling, which means multiple CPU's are available to handle processes at the same time.
2. In a real-world operating system, we should also consider multithreading of CPU to execute multiple processes or threads concurrently.
3. Also, we don't consider process abort in this simulation.