My student Id is : **16270796** so we have **96** processes (that's my luck lol)

I chose these five **CPU** scheduling :

1.First Come First Serve (**FCFS**)

2.Shortest Job First (**SJF**)

3.Longest Job First (**LJF**)

4.Shortest Remaining Time First (**SRTF**)

5.Highest Response Ratio Next (**HRRN**)

(I did my coding with C++ (that's my preference) and each of my coding I did 96 processes randomly generated and my compiler was CodeBlock (In case you need it)).

Comparison of Results :

Algorithm	Avg. Waiting Time	Avg. Turnaround Time	Throughput	Preemptive	Starvation Risk
FCFS	High	Moderate	Moderate	No	Low
SJF	Low	Low	High	No	High
LJF	High	High	Low	No	Low
SRTF	Very Low	Low	High	Yes	High
HRRN	Balanced	Balanced	Moderate	No	Low

Conclusion :

- **FCFS**: Simple but not efficient for minimizing waiting time. It's fair in a sense that processes are served in the order they arrive, but it can lead to inefficiencies, especially when long processes arrive early.
- **SJF**: Highly efficient in terms of through put and waiting time, but suffers from the **starvation problem**, where long processes might never get a chance to execute.
- **LJF**: Not ideal for systems that need responsiveness, as shorter jobs are delayed significantly. This algorithm can be useful when prioritizing long jobs that need uninterrupted execution.
- **SRTF**: The best algorithm for minimizing waiting time but there is a risk of starvation for long jobs.
- **HRRN**: This algorithm has a balanced starvation compare to SJF and SRTF, That's why it's more complex to implement.

  Each of these algorithms has its strengths and weaknesses depending on the context of use. For environments where process burst times are known, SJF or SRTF could be ideal.

  In systems requiring fairness, HRRN offers a more balanced approach.

Event Scenario :

Imagine we have 96 processes already scheduled, and at time **T = 25**, a **new process (P97)** arrives with the following attributes:

- **Arrival Time**: 25
- **Burst Time**: 2 (a very short burst time, requiring immediate attention)
- **Priority**: This new process is urgent and ideally should be completed as soon as possible.

Summary of Reactions to Transient Event :

Algorithm	Reaction to Transient Event	Response Time for P97	Throughput Impact	Overall Suitability
FCFS	No priority for new process	Very High	Reduced	Poor
SJF	Prioritizes P97 after current process finishes	Low	Improved	Moderate
LJF	Delays P97 significantly	High	Reduced	Poor
SRTF	Immediately preempts current process	Minimal	High	Excellent
HRRN	Fairly balances waiting time and burst time	Moderate to Low	Balanced	Good

Conclusion :

When a high-priority, short-burst process arrives suddenly, **SRTF** handles it best.
SJF also handles it well but suffers from being non-preemptive,
HRRN provides a balanced response but may still delay the new process if other tasks have waited too long.
FCFS and **LJF** perform poorly in handling transient events.