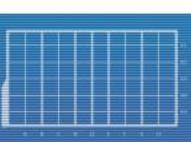
- ► Every scheduling algorithm is different and may favor (boost CPU performance in the presence of) certain types of processes.
- Note: Create your own input test file with several test cases to show which criteria does a particular algorithm is in favor and its disadvantages.



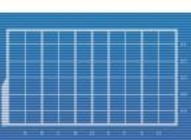




# **Group Project #1**

- ► To do your first group project (by pairs), you need to learn about scheduling algorithms.
- ► CPU scheduling involves looking at the ready queue and picking one (or more, if more than one processor is free) to be run.

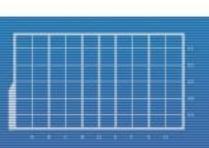






- ➤ You are given an input file. All numbers are integers, no fractions.
  - ► First line: A single number indicating number of test cases.
  - ► Second line: Start of first test case.







- ► For each test case, at least 2 lines:
  - ► First line: A number X greater than zero indicating number of processes, followed by a string: FCFS, SJF, SRTF, P, or RR.
    - ▶In the case of RR, another number Q greater than zero will follow.
  - Next X lines: The X processes. Note that the first process' index is 1. Last process' index is therefore X and not X-

- ► For each process, 1 line, 3 numbers:
  - ➤ First number: Arrival time in ns, always zero or positive (test case start time assumed to be 0ns but processes may not have arrived yet). Assume 5ns New to Ready state transition.
  - ➤ Second number: Burst time in ns, always greater than zero.
  - ► Third number: Priority, range is -20 to





- ► The string after X corresponds to the abbreviation of the algorithm to use.
  - ► FCFS = First Come First Served
  - ► SJF = Shortest Job First (non-preemptive)
  - ➤ SRTF = Shortest Remaining Time First (SJF preemptive)
  - ► P = Priority (preemptive),





- ► Assume a uniprocessor and zero overhead (context-switching is instant).
- ► If there are any "ties", prioritize the process with the smaller index.







- ► Create your own test case
- ► For each test case, output a text version of the resulting Gantt chart:
  - ► First line: Test case number (start with 1).
  - ➤ Second line: First "block", corresponds to the first process to be run.

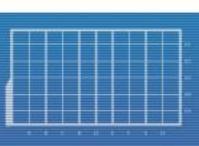






- ► For each block:
  - ► First number: Time elapsed so far in ns.
  - ▶ Second number: Process index.
  - ► Third number: CPU time used in ns. Add a capital X immediately at the end of this number if the process has completed at the end of this block.







# Sample #1

```
2
```

4 SRTF

0 50 2

40 2 3

20 3 1

30 55 1

2 FCFS

100 10 1

10 70 1

▶ Input (via standard input)
▶ Output (standard output)

0 1 20

20 3 3X

23 1 17

40 2 2X

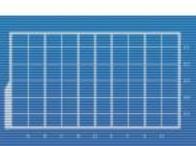
42 1 13X

55 4 55X

2

2 70X

100 1 10X





### Sample #2

► For round-robin, the behavior must mimic FCFS except that preempted processes are moved to the tail end of the queue and, if not currently running, must give way to new arrivals.

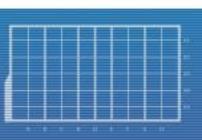
#### ► Input

```
1
4 RR 25
0 30 3
25 45 4
75 10 1
55 15 5
```

#### Output

```
1
0 1 25
25 2 25
50 1 5X
55 4 15X
70 2 20X
90 3 10X
```







- ► What we want to maximize:
  - ► CPU utilization: Must keep CPU busy. Must range from 40% (lightly loaded system) to 90% (heavily used)
    - ▶ 100% might indicate an overloaded system.
  - ► Throughput: Number of processes completed per unit time.
    - Pipelining and multiple processors can only get you so far.



- ► What we want to minimize:
  - Waiting time: Total or sum of times waiting in the ready queue.
    - ► Has NOTHING to do with waiting for I/O.
  - ► Turnaround time: Time it takes to execute the process, from submission (entry into the system) to completion (termination).
    - ► Not the best criterion for an

interactive system. Why?



- ► What we want to minimize (continued):
  - ➤ Response time: Time it takes to start responding, from submission to first response.
    - ► Some output can be produced early, after all.
    - ► But this does NOT include the time it takes to actually output the ENTIRE response.
    - Generally limited by the speed of the output device.



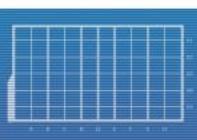
## Scheduling Criteria Output

- Compute the 5 scheduling criteria/measure for each case and scheduling algorithm
- ➤ Output a comparison of performance for all scheduling algorithms similar to the table in <u>03.06 (Fri) CPU Scheduling Algorithm</u> <u>Performance.</u>
  - CPU Utilization: (%), set a fixed clock cycle time for system on and off
  - Throughput: (# of processes) completed per unit of time
  - Waiting time: (time) cumulative
  - Turnaround time: (time)
  - Response time: (time)

CPU	Average*	Average*	Average*	Throughput	CPU Utilization*	CPU Utilization*
Scheduling	Response Time	<b>Waiting Time</b>	Turnaround Time	on the 13th cc	on the 36th cc	on the 25th cc
Algorithm	(clock cycles)	(clock cycles)	(clock cycles)	(# of processes)	(%)	(%)
FCFS						
SJF						
RR						

<sup>\*</sup> use 2 decimal places format







#### Submission and Deadline

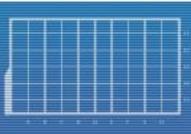
Submit the following files in Moodle <u>03.23 (Mon) - Project #1</u> using the following filenames / formats below. Do not zip the files.

- CE155\_Project1\_<Surname1>\_<Surname2>\_code.cpp
- CE155\_Project1\_<Surname1>\_<Surname2>\_testfile.txt
- CE155\_Project1\_<Surname1>\_<Surname2>\_output.pdf
- CE155\_Project1\_<Surname1>\_<Surname2>\_readme.txt
- CE155\_Project1\_<Surname1>\_<Surname2>\_video.mp4

Read me file should show a short description of the program — what is the program, how to run and use the program, its functions and operations.

Submit your demo videos in: CE155 Project #1 GDrive Submission Folder.







### **Demo Video**

- ➤ 7 mins demo video to present your code and output
- Moodle Activity will be posted for submission of:
  - ► C/C++ code and executable file
  - ► Test file and output file
  - ▶ Video



