Disclaimer: This could be misleading, use with caution. If other resources help, please do not be discouraged to use them. This is merely my take on the assignment.

Main Idea : Create a process scheduling program

We need a way of **organizing** the tasks. This will be done using a **priority** queue.

We will call this the **MainQueue**.

Inside this main queue:

- We should only have **arrival times** and **completion times** for a process
- **Arrival Times** are loaded **before** the MainQueue loop begins. So all the arrival times of **all** the processes are loaded in.
- **Completion Times** is time a certain process needs to complete its command (INPUT,CPU,SSD). This will be the sum of the clock time and time needed. In other words: **Completion Time** = **clock time** + **time needed.**
- Below this is indicated as **#1** in the **object**

With this knowledge lets create the initial MainQueue (only arrival times):

Input	
BSIZE 4096 **START 0** CORE 200 WRITE 4096 CORE 30 **START 100** CORE 80 WRITE 4096 CORE 40	We have two process by the looks of it: 1. START 0 2. START 100 We need to add these to an **object** and have it store three subfields: 1. Time (in this case arrival) 2. Operation (in this case START) 3. Process# 4. The number of logical_reads 5. The number of physical_reads 6. The number of physical_writes (Note 4-6 are not shown below) We can make these process below in the MainQueue:

Step 1: Add the process to the MainQueue (**Left**: Most Important (**smallest time** value)

1. 0ms | 2. START | 3. 0	1. 100ms | 2. START | 3. 1

Now our MainQueue is done! We need to start the while loop:
Step 2: Make sure you have a **clock variable** set to **0 before** the while loop.

- Clock = 0

Step 3: Start the while loop

While the MainQueue is not empty:

Pop the top element and store it

a. Our top element in this case would be → **0**ms | 2. START | 3. 0 1
b. Our MainQueue now looks like

100ms | 2. START | 3. 1

Set the clock time to the time of the top element

a. Clock = **0**

If it an arrival:

b. We will have to schedule its next event (it will always be core)

Based on the input chart above :

START 0
CORE 200 → this is the process to schedule.

We push this back into the MainQueue like so:

- TimeCompletion = time need + clock time = 200 + 0 = 200
- In other words, this process will complete at time **200** in the **core**
- So we push this **information** (completion time) back into the MainQueue

1. **100**ms | 2. START | 3. 0	1. **200** ms | 2. Core | 3. 1

Step 4: Start the while loop again (MainQueue is not empty still)

While the MainQueue is not empty:

1. Pop the top element and store it
    a. Our top element in this case would be → **100**ms | 2. START | 3. 0 1
    b. Our MainQueue now looks like

1200 ms | 2. START | 3. 1

Set the clock time to the time of the top element

c. Clock = **100**

If it an arrival:

d. We will have to schedule its next event (it will always be core)

Based on the input chart above :

START 100

CORE 80 → this is the process to schedule.

We **cannot** push this back into the MainQueue!

- Why? The core is currently **full** with process 1 till its completion at **200**ms
- This process must go into the **Ready Queue**
- **Ready Queue is**
-

1. 80 | 3. Process 0

    - Note: We only need to know the time it needs and the process because its in a ready queue so it must be a cpu queued event (reason there is only two elements)

Step 5: Start the main queue again (still not empty)

1. **200** ms | 2. CORE | 3. 1

1. Pop the top and store the element
    a. Our top element in this case would be → **200** ms | 2. CORE| 3. 0 1
    b. Our main queue is empty (but we are not done)
    c. Set Clock time = 200
2. Since this is a core completion event
    a. We need to check if the RQ has an element that needed to be served

1. 80 | 3. Process 0

It does. Pop it and add it to the mainqueue but first

We need its **completion time** to put it in the mainQueue.
Completion Time = time needed + clock time = 80 + 200 = 280

So now our MainQueue is:

1. 280 ms | 2. CORE | 3. 0

3. Now we need to process our Core Completion to its next event:

CORE 200 → Just finished (at time 200)
WRITE 4096 → (this is its next event)

Do the same process, follow the python code.