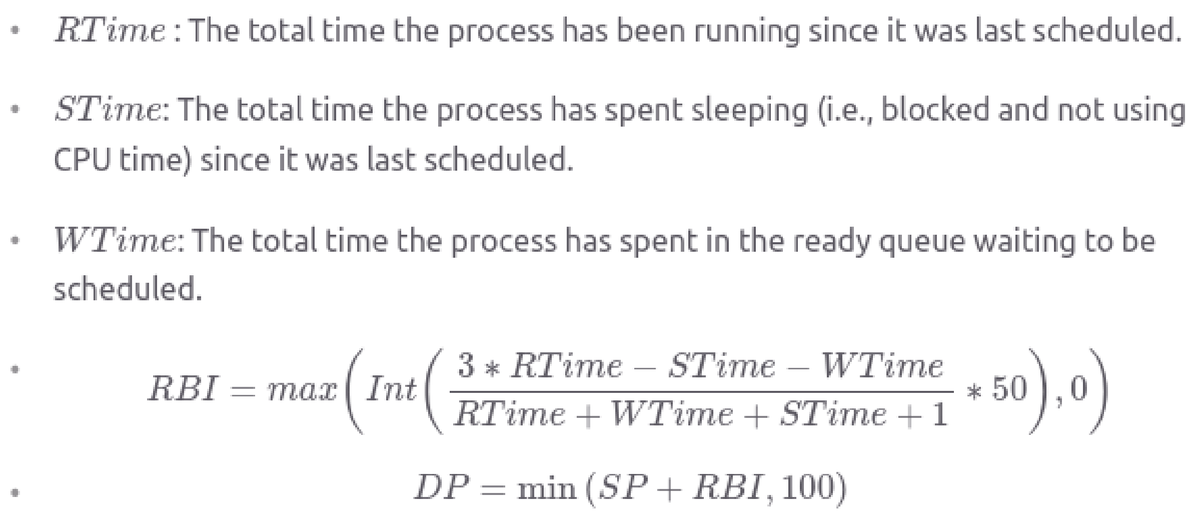
Mini Project – 3
Priority Based Scheduler

1. Code Explanation

- Modify the struct proc, to include RBI, Rtime, wtime, stime, dp, sp, sched [number of times scheduled], where:
  RBI (Recent Behaviour Index) and dp (dynamic priority) are given by :



- $RTime$: The total time the process has been running since it was last scheduled.
- $STime$: The total time the process has spent sleeping (i.e., blocked and not using CPU time) since it was last scheduled.
- $WTime$: The total time the process has spent in the ready queue waiting to be scheduled.
- $$RBI = max\left(Int\left(\frac{3 * RTime - STime - WTime}{RTime + WTime + STime + 1} * 50\right), 0\right)$$
- $$DP = \min\left(SP + RBI, 100\right)$$

- In the allocproc() function, initialize RBI to its default value of 25, sp to 50 [=> dp = 75] and other to 0
- In the scheduler() function, iterate over all the processes in the proc array, store the details of the process with minimum dp and minimum sched [in case of same minimum dp] and minimum ctime [incase of same minimum dp and minimum sched].
- Schedule this process
- For setpriority system call, iterate over all the processes in the proc array to find the process with given pid, for this process, update the sp value, to the given sp [before that, store the previous sp]. Now calculate the new dp value and store it in dp field of the proc struct. Reset RBI to 25. Check if the dp values [new] is less than the previous dp value, if so; call the yield() function, to reschedule the process, else; return the old sp value;
- ASSUMPTIONS made for the setpriority syscall:
  - Since it is not mentioned clearly whether calculation of new dp after changing sp before or after resetting RBI to 25; here I assumed that before resetting RBI, dp is calculated, then RBI is set to 25 and then if the dp value is lower than before, yield function is called.

2. Analysis

- Static Priority (SP):
  - Reduced SP levels signify more priority
- Recent Behaviour Index (RBI):
  - Lower values of RBI indicates that a process has been waiting or sleeping more than it ran, leading to a lower dp
  - Higher RBI indicates a process has been actively using the CPU, leading to higher dp

Trivially, since we will be setting RBI to 0 incase the formula leads to a negative value, and setting to 0 reduces the dp, increasing the priority, it can be observed that RBI is essential for handling starvation [because as a process's wait-time increases, means it starvates, RBI value decreases and becomes negative]. Because of this reason, RBI is trivially a better scheduling algorithm than Round Robin, and FCFS.

Cafe Sim

1. **Implementation details**
   - The time(s) taken to prepare coffee(s) is/are stored in a hashtable
   - As soon as a customer arrives, a thread is creadted corresponding to that customer
   - In this thread, the customer waits for his turn to arrive [similar to tickets and turns lock idea discussed in class], as we need to ensure completion of cust i order before cust i+1
   - If it is the customer’s turn, wait for a barista to be freed
   - When a free barista is found, check if the order can be finished before tolerence time of the customer. If so, sleep for preparation time and print the message that barista finished preparing the coffee. Else, sleep for remaining time [cust tolerance time – cust waiting time], print the message that the customer left, sleep for the remaining time, for the coffee to be completed
2. **Waiting Time**
   - In the given example, customer 1 arrives at o second(s), his/her order is started at 1 second(s). Hence his waiting time is 1 second.
   - Customer 2 arrives at 3 second(s), his/her order is started at 4 second(s). Hence his waiting time is 1 second(s).
   - Customer 3 arrives at 3 second(s), his/her order is started at 8 second(s). Hence his waiting time is 5 second(s)
   - Average waiting time in this case is: (1+1+5)/3 = 2.33 second(s)
   - In general, average time is calculated as :
     (Sum of (coffee preparation start time – coffee ordered time))/Number of customers
   - In case cafe had infinite baristas, since for every customer, a barista becomes available in the next immediate second, average waiting time is 1 second
3. **Coffees Wasted –** Included in code

Ice Cream Parlor

[Since it is not explicitly mentioned in the question that if multiple customer's orders are possible at a given time, the lowest customer should get priority, it is assumed that in this case, orders are taken non-deterministically]

1. Implementation details:

- Each customer, machine and order of each customer are taken as threads
- In the customer function, check if there empty machines left. If so, check if any machine can accept and finish the order. If so, start the order_thread.
- At any point of time, if all machines have stopped working, print the corresponding message and return

2. Minimizing Incomplete Orders:

- One possible greedy approach is to prioritize orders based on the number of orders per customer. So that, the customer with lesser number of orders is finished/completed first, effectively reducing the number of incomplete orders
- However, this greedy algorithm may not work always.

3. Ingredient Replenishment:

- Whenever a particular ingredients quantity drops below a threshold value, say 2 or 3 [exact perfect value depends on case to case], the ingredients can be replineshed.
- For this, a possible solution is to create a separate thread, similar to say an employee of the parlor, whoose responsibility to continuously manage the inventory of the ingredients/toppings, and order the required ingredients in bulk [say 10 or 20 at a time] from the supplier, whenever an ingredient's quantity drops below the threshold

4. Unserviced Orders:

- A greedy approach might be to take orders that take least time to complete first, however this might not always work [in cases such as:-
    1. Vanilla – Caramel -> 1 second
    2. Vanilla - Brownie, Caramel -> 1 second
    3. Vanilla – Brownie -> 1 second
    4. Chocolate – Brownie -> 2 seconds
    5. Chocolate – Brownie -> 2 seconds
    6. Chocolate – Caramel -> 2 seconds
    7. Chocolate – Caramel -> 2 seconds

Consider there is only 1 machine, that operates from 0 to 50 seconds and that Caramel and Brownie are available in quantities – 3 each. If least time to complete is considered, 1, 2, 3 are only served; where as 4, 5, 6, 7 is also a possible combination]