Preemptive Priority-Based Scheduling

Part A

My output is:

```
xenomai@ieu:~/exercises/preemptive$ ./ex04a
start task  : 0
Task  : 0
start task  : 1
Task  : 1
start task  : 2
Task  : 2
wake up all tasks
Running Task  : 2  at ms : 10
.
.
.
Running Task  : 2  at ms : 200
End Task  : 2
Running Task  : 1  at ms : 10
.
.
.
Running Task  : 1  at ms : 200
End Task  : 1
Running Task  : 0  at ms : 10
.
.
.
Running Task  : 0  at ms : 200
End Task  : 0

Type CTRL-C to end this program
```

Task 2 executes first and task 0 executes last as expected. Each task spends 200 ms waiting burning CPU cycles (so nothing).

Part B

This is my output:

```
xenomai@ieu:~/exercises/preemptive$ ./ex04b
start task  : 0
Task  : 0
start task  : 1
Task  : 1
start task  : 2
Task  : 2
wake up all tasks
Running Task  : 1  at ms : 10
.
.
.
Running Task  : 1  at ms : 200
End Task  : 1
Running Task  : 2  at ms : 10
.
.
.
Running Task  : 2  at ms : 200
End Task  : 2
Running Task  : 0  at ms : 10
.
.
.
Running Task  : 0  at ms : 200
End Task  : 0

Type CTRL-C to end this program
```

Task 1 and 2 now have the same priority that's higher than task 0. Task 0 still executes last but task 1 now executes before task 2. I suspect that task 1 executes first because it is started before task 2. Even though each task is signaled to execute at the same time by rt_sem_broadcast, priority then seems to default the task that is started first. So round-robin scheduling or FIFO scheduling?

Part C

Here's the truncated output of my script:

```
xenomai@ieu:~/exercises/preemptive$ ./ex04c
start task  : 0
Task  : 0
start task  : 1
Task  : 1
start task  : 2
Task  : 2
wake up all tasks
Running Task  : 2  at ms : 10
.
.
.
Running Task  : 2  at ms : 100
Running Task  : 2  at ms : 110
Running Task  : 1  at ms : 10
.
.
.
Running Task  : 1  at ms : 200
End Task  : 1
Running Task  : 0  at ms : 10
.
.
.
Running Task  : 0  at ms : 200
End Task  : 0
Running Task  : 2  at ms : 120
.
.
.
Running Task  : 2  at ms : 200
End Task  : 2
```

I had task 1 and 0's priorities reassigned when task 2 had a runtime = 110 ms. Here, you can see that when task 1 and 0's priorities were raised by 10, they were higher than task 2's. At this point, the priorities are

Task 0 = 60
Task 1 = 61
Task 2 = 52

Task 2 is immediately preempted by task 1 once this reassignment occurs. Task 1 starts and runs to completion then task 0 begins to execute. Task 0 runs to completing and then task 2 resumes execution. Preemptive priority-based scheduling

Part D

Here's my truncated output:

```
xenomai@ieu:~/exercises/preemptive$ ./ex04d
start task  : 0
Task  : 0
start task  : 1
Task  : 1
start task  : 2
Task  : 2
wake up all tasks
Running Task  : 2  at ms : 10
.
.
.
Running Task  : 2  at ms : 100
Running Task  : 2  at ms : 110
Running Task  : 1  at ms : 10
.
.
.
Running Task  : 1  at ms : 100
Running Task  : 1  at ms : 110
Running Task  : 0  at ms : 10
.
.
.
Running Task  : 0  at ms : 200
End Task  : 0
Running Task  : 1  at ms : 120
.
.
.
Running Task  : 1  at ms : 200
End Task  : 1
Running Task  : 2  at ms : 120
.
.
.
.
```

```
Running Task  : 2  at ms : 200
End Task  : 2
```

Halfway through its execution, task 2 reassigns its priority to the lowest out of all three. Once it does so, task 1 preempts task 2. Halfway through its execution, task 1 reassigns its priority such that its lower than task 0 but higher than task 2. Once it does so, task 0 preempts task 1. Then, task 0 executes and completes. Once it completes, task 1 begins to execute again and continues from where it left off. Once complete, task 2 does the same thing.

Essentially, priorities were inverted.