Nome: Thaila Schmidt

Para o trabalho 2, foi renomeado o scheduler do trabalho 1 para scheduler_loteria() e o do trabalho 2 para scheduler_stride(), assim o scheduler() é quem decide qual dos dois irá executar.

O comando para executar o trabalho 2 é **testS**.

A seguir, um exemplo da saída esperada do programa (sendo 0, 1 e 2 as ids), com aqueles valores de bilhetes que estão no código:

0	1	2
0	0	0
0	0	40
0	200	40
100	200	40
100	200	80
100	200	120
200	200	120
200	200	160

ETC.

Logo, os programas escalonados teriam a ordem **21022022…**

Agora um print da saída do programa:

OBS: Deixei vários prints para entender como ele está funcionando, logo serão explicados. Em algum momento o código “se perde”, eu não sei ao certo o motivo.

Para entender os prints:

O *min pass* está verificando qual é o processo com menor passo, ao encontrar o menor, ele faz *stop* no mesmo e depois vem a *soma*, que irá somar os passos correspondentes ao processo executado.

Então a ordem deveria ser min pass -> stop -> soma, porém as vezes ele pula o *stop*.

```
min pass=0 pid=4
min pass=0 pid=5
min pass=0 pid=6

stop pid=6 (id=2)

soma do passo: 40
min pass=0 pid=4
min pass=0 pid=5

stop pid=5 (id=1)

soma do passo: 200
min pass=0 pid=4

stop pid=4 (id=0)

soma do passo: 100
min pass=40 pid=6
soma do passo: 80
min pass=100 pid=4
min pass=80 pid=6

stop pid=6 (id=2)

soma do passo: 120
min pass=100 pid=4

stop pid=4 (id=0)

soma do passo: 200
min pass=200 pid=5

stop pid=5 (id=1)

soma do passo: 400
min pass=200 pid=4
min pass=120 pid=6

stop pid=6 (id=2)

soma do passo: 160
min pass=200 pid=4

stop pid=4 (id=0)

soma do passo: 300
min pass=400 pid=5

stop pid=5 (id=1)
```

Aqui podemos ver que os primeiros processos funcionaram de

acordo com o esperado, porém, por algum motivo ele não fez *stop* no 2 ao somar 80, ele também acaba identificando o 1 como menor e somando 400 quando deveria ser o 2, mas depois volta a executar corretamente.

Geralmente acontecem alguns erros desse tipo, mas nem sempre são os mesmos erros, deve ser a lei de Murphy…

(no início terão vários prints com valor 0, mas depois ele inicia)