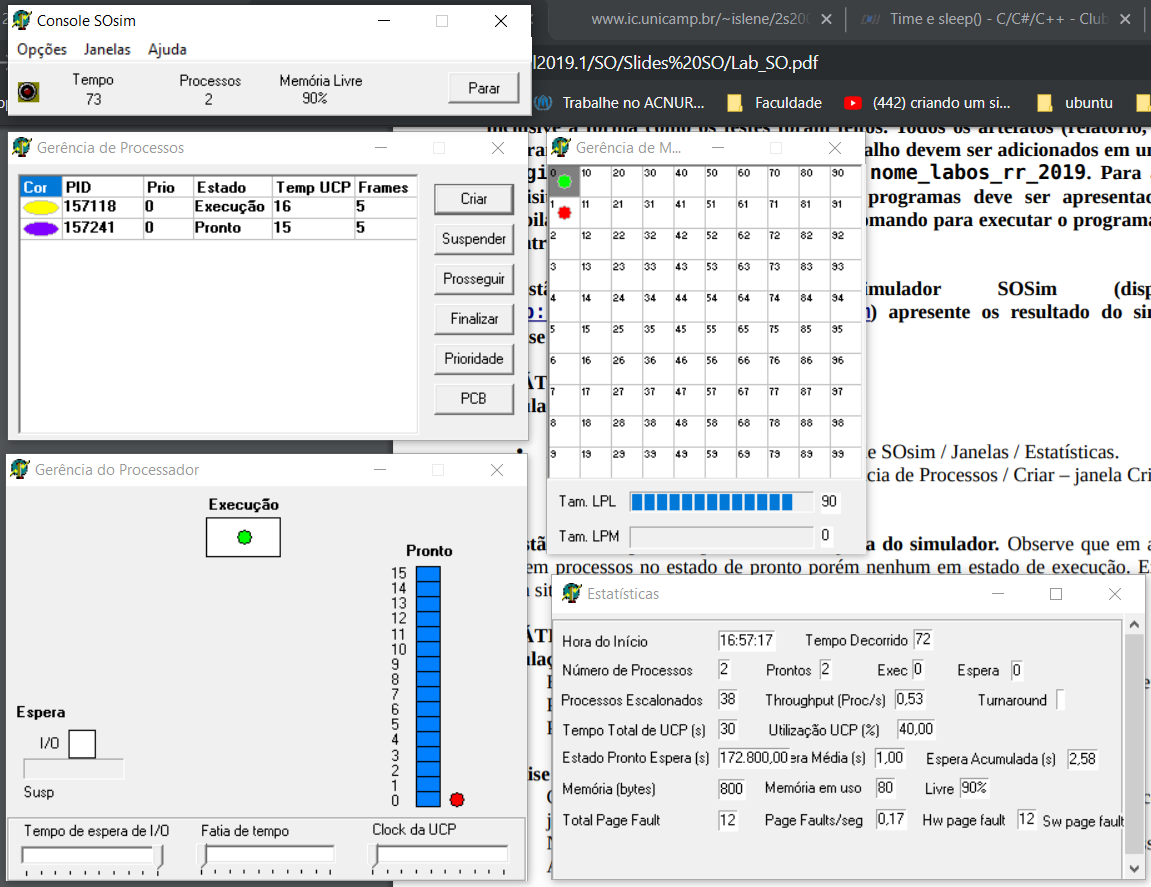
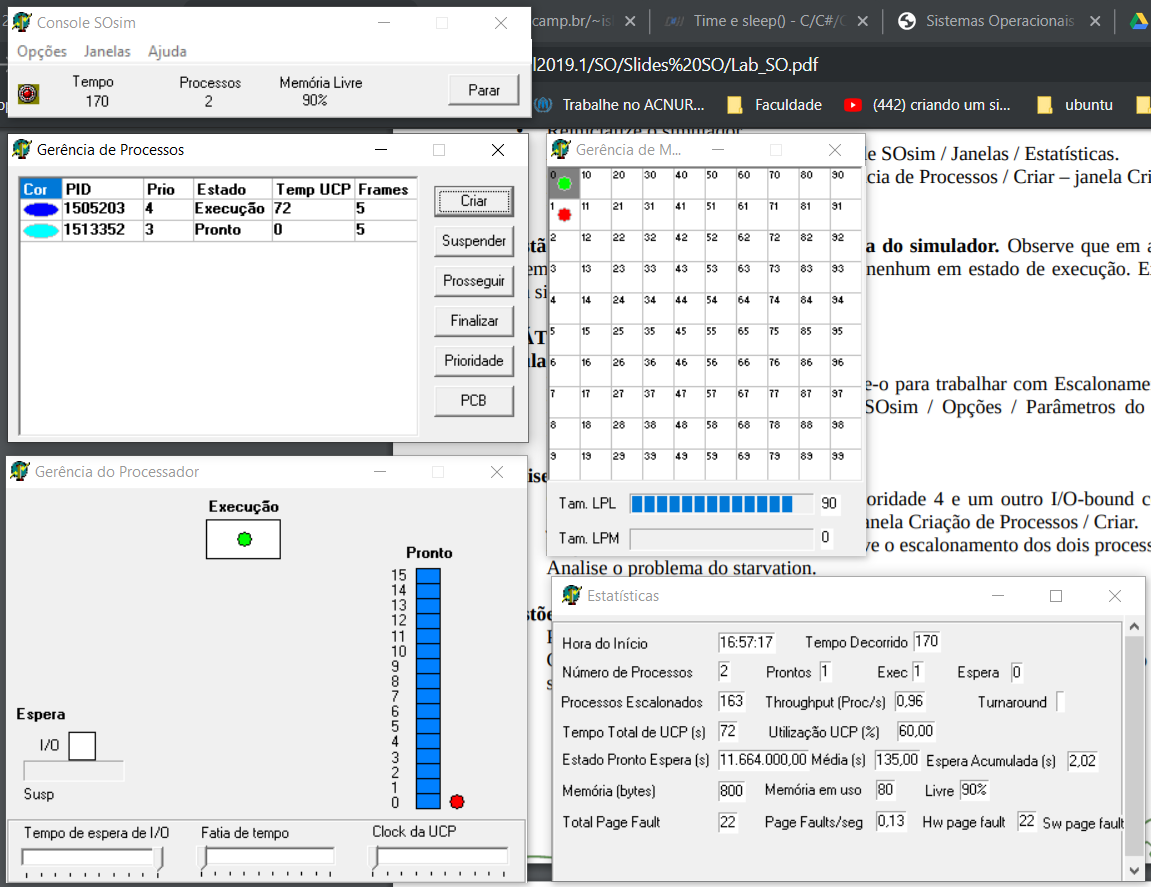
**Relatório**

**Questão 1:**

**Prática A:**

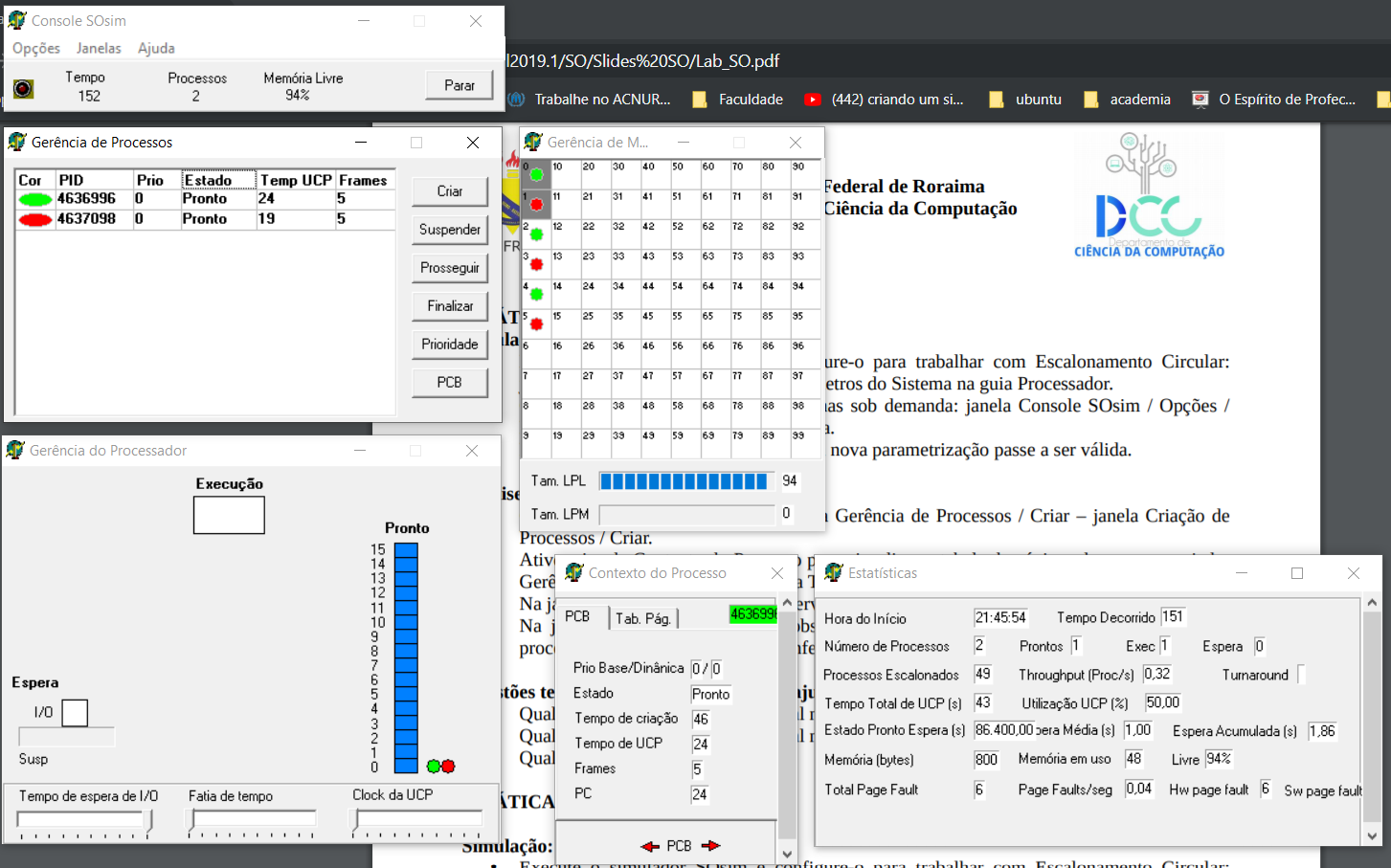
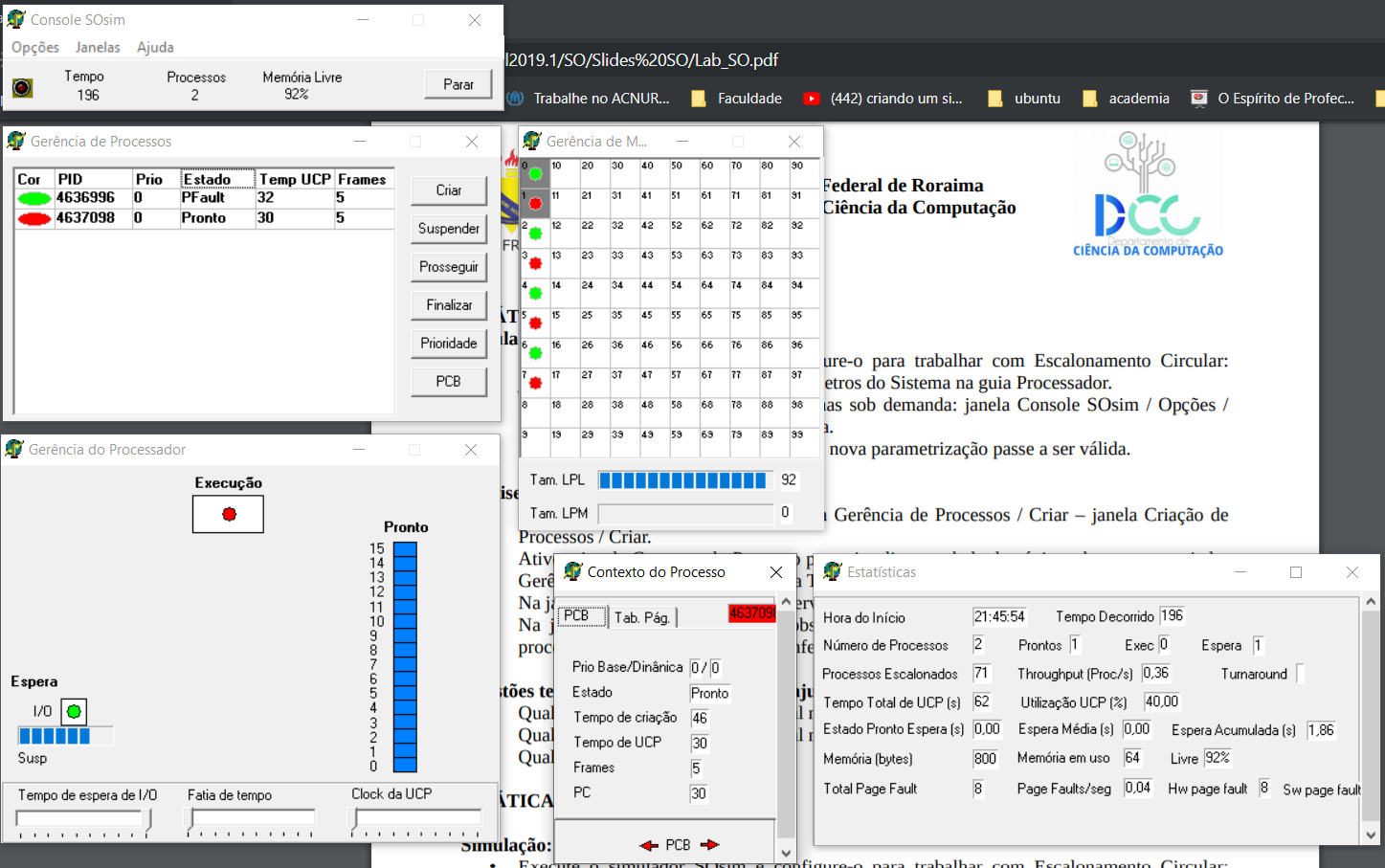
Os processos estão sendo executados através de escalonamento circular, ou seja, eles seguem uma ordem de execução. Nesse tipo de escalonamento cada processo tem um tempo para executar na cpu, se não terminar no seu tempo, ocorre um delay devido a mudança de contexto para outro processo, o conteúdo do processo em execução é salvo, e o outro processo que antes esperava é carregado, passando ao estado de execução.

**Prática B:**

Ocorre starvation quando processos de maior prioridade concorrem pelo kernel contra processos de menor prioridade. No exemplo cpu-bound tem prioridade 4 e i/o-bound prioridade 3, assim o processo com maior prioridade fica com acesso ao kernel, já o processo com menor prioridade nunca tem permissão, assim fica parado, sem fazer nada ocupando memória e tempo de processamento.

Reduzir a prioridade ou finalizar o processo que gerou o starvation. Ou poderíamos igualar a prioridade do processo que estava em starvation.

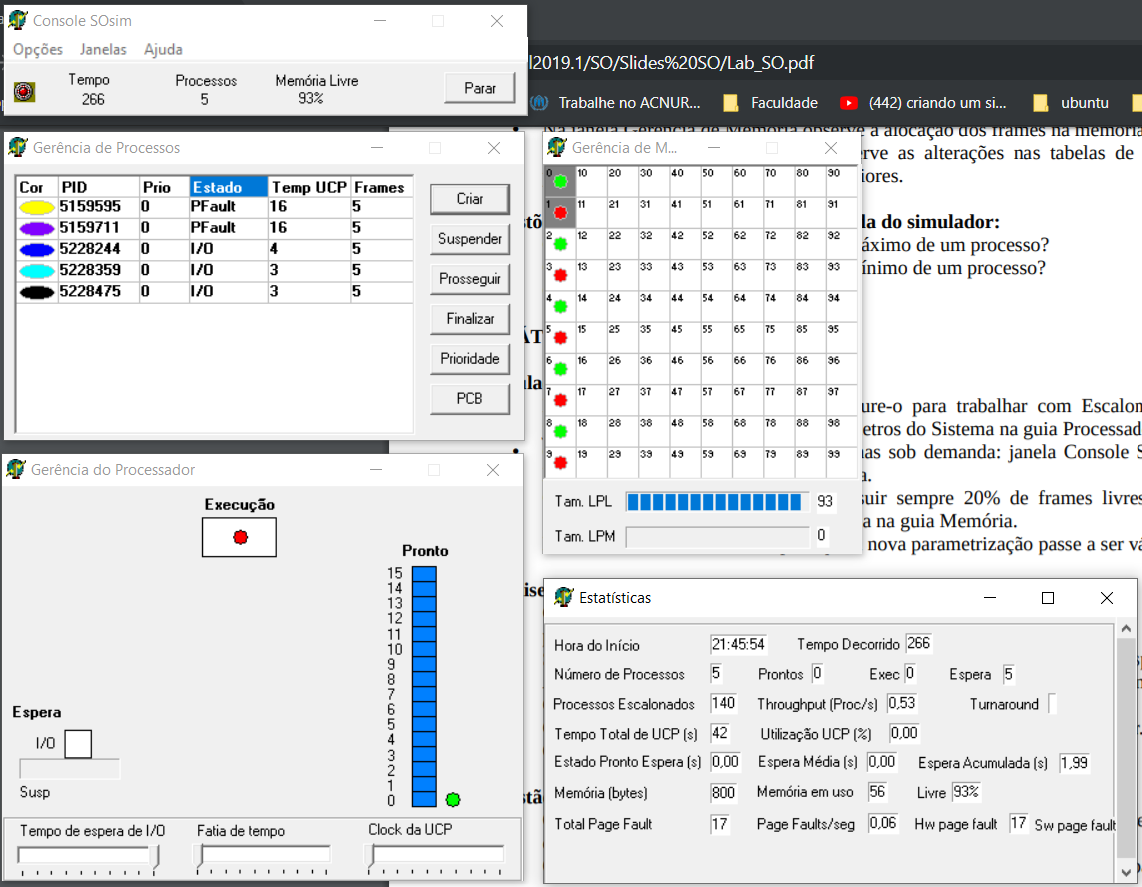
**Prática C:**

****

O espaço de endereçamento máximo é o total da RAM + o total do disco rígido, ou seja, é toda a memória primária e secundária.

O espaço real mínimo de endereçamento é tamanho mínimo da tabela de mapeamento carregada.

O tamanho típico de uma página é de 4 [KiB,](https://pt.wikipedia.org/wiki/KiB) porém isso não é uma regra, conforme a arquitetura de hardware e diferentes SO’s esse valor muda.

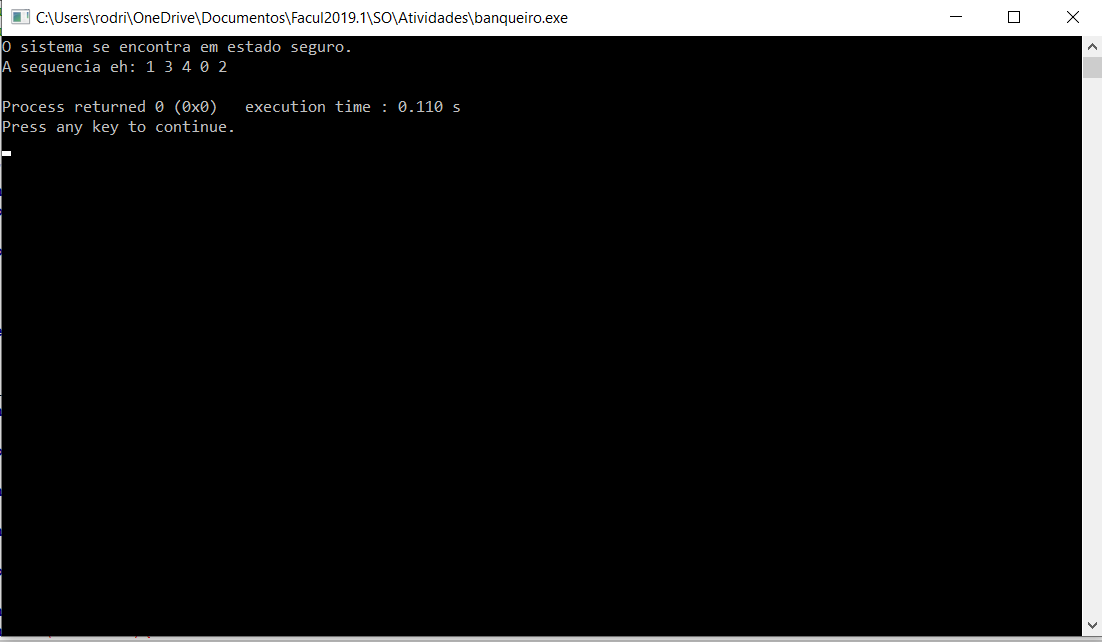
**Prática D:**

O simulador usa o critério onde dá swap-out no processo com menor chance de entrar na CPU.

Quando o processo requer um dado, mas esse dado não está na memória, assim o dado é trazido da memória secundária

**Questão 2:**

O Algoritmo do banqueiro é um algoritmo de alocação de recursos com prevenção de [impasses](https://pt.wikipedia.org/wiki/Deadlock) desenvolvido por [Edsger Dijkstra](https://pt.wikipedia.org/wiki/Edsger_Dijkstra" \o "Edsger Dijkstra) que testa a segurança pela simulação da alocação do máximo pré-determinado possível de montantes de todos os recursos computacionais, e em seguida faz uma verificação de estados-seguros para testar a possibilidade de condições de [impasse](https://pt.wikipedia.org/wiki/Deadlock) para todas as outras atividades pendentes, antes de decidir se a alocação deve prosseguir. Esse nome foi escolhido por tratar do problema através da seguinte analogia: um banqueiro (o Sistema Operacional) de uma pequena cidade pode negociar com um grupo de clientes (os processos), aos quais ele libera linhas de crédito (que são recursos do sistema). O algoritmo tem como função determinar se a liberação de uma requisição é capaz de levar o sistema a um estado inseguro, se for este o caso, o banqueiro nega a requisição.

**Exemplo de saída esperada**

**Questão 3:**

Na barbearia há um barbeiro, uma cadeira de barbeiro e *n*cadeiras para eventuais clientes esperarem a vez. Quando não há clientes, o barbeiro senta-se na cadeira de barbeiro e cai no sono. Quando chega um cliente, ele precisa acordar o barbeiro. Se outros clientes chegarem enquanto o barbeiro estiver cortando o cabelo de um cliente, eles se sentarão (se houver cadeiras vazias) ou sairão da barbearia (se todas as cadeiras estiverem ocupadas). A solução aqui apresentada usa três semáforos: CLIENTES, que conta os clientes à espera de atendimento (exceto o cliente que está na cadeira de barbeiro, que não está esperando); BARBEIROS*,* o número de barbeiros (0 ou 1) que estão ociosos à espera de clientes, e mutex, que é usado para exclusão mútua.

**Exemplo de saída esperada**

