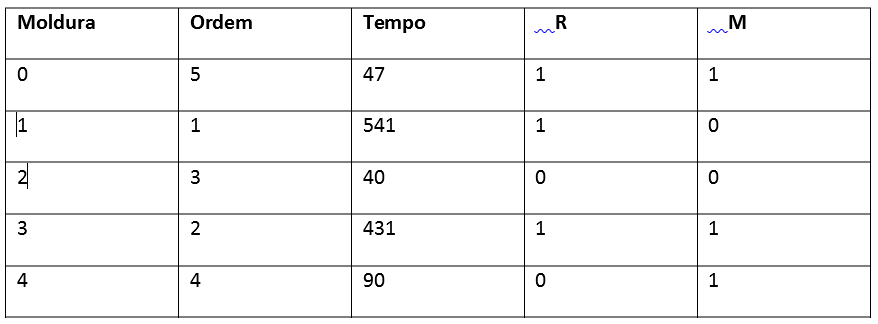
**Infraestrutura de Software - 2022.2 - Final**

**1**. O núcleo (*kernel*) de um sistema operacional pode ser estruturado de formas distintas, como, por exemplo, usando a abordagem monolítica ou micronúcleo (*microkernel)*. Explique a estrutura micronúcleo e cite vantagens em relação à estrutura monolítica. (2,0)

2. Um sistema operacional possui estruturas de dados para gerenciar o contexto de processos e *threads*. Explique as informações que são guardadas respectivamente em um contexto de processos e *threads*. (2,5)

**3.** A tabela abaixo demonstra uma memória com 5 molduras de páginas. A coluna Moldura representa a identificação da moldura.;Ordemindica a ordem de inserção das páginas na memória (o menor número, inserção mais cedo. Ex: moldura 1 tem a primeira página adicionada na memória); Tempo apresenta a última referência feita a página alocada na moldura; os bits R e M representam, respectivamente, referenciado e modificado. Assuma o tempo atual 600. Para os seguintes algoritmos, qual a moldura terá a respectiva página substituída: (i) FIFO, (II) Segunda chance; (III) Não recentemente usada; (iv) Conjunto de Trabalho, assumindo a constante ᵼ = 550. Por fim, descreva brevemente o algoritmo WSClock – Conjunto de Trabalho Relógio (2,5)  
 ****

5. Em um sistema de controle de estoque, deseja-se fazer o levantamento da quantidade de produtos armazenados. Para isso, faça um programa que irá ler um número N de arquivos com as informações sobre o estoque. Cada arquivo terá um produto por linha que será um número 1 <= y <= 10, representando um dos 10 tipos de produtos estocados. Threads serão utilizadas para fazer a contagem da quantidade, lendo os dados dos arquivos e incrementando um vetor *qtde\_prod* de tamanho 10.. Você deverá garantir a exclusão mútua separada para cada posição do array *qtde\_prod*. Mais especificamente, enquanto um produto *x* está sendo contabilizado por uma *thread* e esta modificar o array na respectiva posição, uma outra thread pode modificar o array em uma posição *y* que representa outro produto. Obviamente, se mais de uma thread quiser modificar a mesma posição do array *qtde\_prod* simultaneamente,somente 1 terá acesso, e as outras estarão bloqueadas.. Faça a implementação utilizando *pthreads*. **Foque no controle da modificação de** *qtde\_prod.* **int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex); int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex); int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex); int pthread\_cond\_signal(pthread\_cond\_t \*cond);int pthread\_cond\_broadcast(pthread\_cond\_t \**cond*); int pthread\_mutex\_init(pthread\_mutex\_t \*mutex, const pthread\_mutexattr\_t \*attr);int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex); int pthread\_join(pthread\_t *thread*, void \*\**value\_ptr*); int pthread\_barrier\_wait(pthread\_barrier\_t \*barrier); int pthread\_barrier\_init(pthread\_barrier\_t \* barrier, const pthread\_barrierattr\_t \* attr, unsigned int count ); (3,0)**