Sistemas de Informação – EACH-USP
Sistemas Operacionais – Prof. Alexandre S. Freire
Prova 1 - 28 de setembro de 2016

LEIA ATENTAMENTE AS INSTRUÇÕES ABAIXO:

- Pode fazer a prova a lápis;
- Escreva seu nome completo e NUSP em todas as folhas, inclusive nesta;
- Devolva todas as folhas, inclusive os rascunhos e esta folha;
- Desligue todos os aparelhos eletrônicos e não converse durante a prova;
- Pode fazer as questões em qualquer ordem;
- Faça apenas o que as questões pedem, sem se preocupar com o restante do código;
- Qualquer dúvida, levante a mão e aguarde o professor chegar até você ou vá até o professor (não fale em voz alta).

Questão 1 (Valor: 2.5). Quais são as 4 condições necessárias para haver a possibilidade de ocorrer um *deadlock*? Além do *deadlock*, existe mais alguma situação "preocupante" que todo escalonador de processos deveria considerar? Se sim, qual? Justifique sua resposta.

Questão 2 (Valor: 2.5). Desenhe o *Diagrama de Gantt* para ilustrar o funcionamento do escalonador *Shortest Job First Preemptivo* (SJFP) (também conhecido como *Shortest Remaining Time First*) para a seguinte situação:

Processo	P_1	P_2	P_3	P_4	P_5
Instante de chegada	0	0	4	9	17
Duração do próximo *burst* de CPU	10	8	3	1	4

Qual é o tempo médio de espera? Há algum outro escalonador que obteria um tempo médio de espera inferior ao SJFP para a mesma situação descrita acima? Se sim, desenhe o *Diagrama de Gantt* correspondente e calcule o respectivo tempo médio de espera.

Questão 3 (Valor: 2.5). Para que serve o *Algoritmo do Grafo de Alocação de Recursos* (AGAR)? Descreva como ele funciona. Construa um exemplo no qual um processo solicita um recurso disponível mas que o sistema operacional, após executar o AGAR, não o aloca para evitar uma possibilidade de deadlock.

Questão 4 (Valor: 2.5). Escreva uma classe `BoundedBuffer` em java, que contenha os métodos `void insert(Object o)` e `Object remove()`, para controlar o acesso a um buffer compartilhado de tamanho k (ou seja, sua classe deve apresentar uma solução para o *Problema do Bounded Buffer*). O parâmetro k deve ser recebido no construtor da classe. Assuma que existe uma classe `Semáforo`, já implementada, que contém os métodos `acquire` e `release`. Para instanciá-la, utilize o comando `new Semáforo(n)`, onde n é o valor inicial do semáforo. **Dica:** se preferir, utilize a classe `LinkedList`, disponível no pacote `java.util`, para armazenar os objetos inseridos no buffer. Os métodos que poderá utilizar da classe `LinkedList` são os seguintes: `void addLast(Object o)`, `void addFirst(Object o)`, `Object removeLast()` e `Object removeFirst()`.