CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca
Unidade Maracanã

Curso	Período	Departamento	Prova
CSTSI ( ) BCC ( )	2º	DEPIN	P1 (X) P2 ( ) PF ( ) 2ªChamada ( )

Disciplina	Professora	Data
Estruturas de Dados	Myrna Amorim	25/09/19

Nome	Nota

Importante:

1. ***Tempo máximo de duração da avaliação: 2 horas.***
2. *Leia atentamente toda a questão antes de iniciar;*
3. *A resposta deverá ser à **caneta** (azul ou preta);*
4. *PROIBIDO USAR O CELULAR DURANTE A PROVA.*
5. *Coloque sempre o número da questão na folha de resposta;*
6. *As questões poderão ser respondidas fora de ordem, desde que seja obedecido o item 5.*

---

1 (2.0). A forma como é realizada a inserção e a remoção dos elementos numa lista linear (estática ou dinâmica) indica a política que está sendo empregada, sendo as mais comuns: pilha (LIFO) e fila (FIFO). Existem diversas aplicações dessas políticas em sistemas computacionais: compiladores, sistemas operacionais, armazenamento de dados para impressão (*spooling*), roteamento entre outros. Utilizando como base o funcionamento de um *spooler* (*spool*) de impressão, onde os documentos ficam armazenados e impressos na ordem de chegada, implemente as seguintes funções utilizando uma lista linear simplesmente encadeada e dois ponteiros (um para apontar para o início e outro para o final da lista), mantendo a complexidade de pior caso O(1) para cada função.

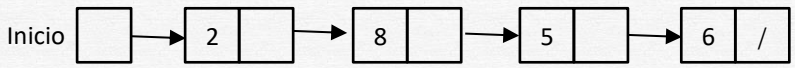
a) inserir no *spooler*: envio do documento na lista.
b) retirar do *spooler*: remoção do documento da lista (lembrando que a impressão é pela ordem de chegada).

Utilize como base a *struct*:

```
struct spool
{
    int num;
    struct spool * prox;
};
```

2 (2.0). Implemente:

a. (0.8) Faça uma função que solicite um número inteiro N, percorra uma lista simplesmente encadeada e ao encontrar N, remova-o da lista. Caso não o encontre, mostre a mensagem "Elemento não encontrado."
b. (0.8) Elabore uma função que percorra uma lista simplesmente encadeada e mostre o nó que possui o maior valor e o nó sucessor. Não se esqueça de verificar a particularidade quando o nó de maior valor for o primeiro ou o último. Usando o exemplo, seriam mostrados os números: 8 e 5, respectivamente.



c. (0.4) A complexidade de pior caso das funções criadas nos itens “a” e “b” são iguais? Justifique, citando inclusive os valores das complexidades.

3 (2.0). Usando como referência a explicação do algoritmo a seguir, faça uma função que ordene um *array* com 20 elementos inteiros. Informe a complexidade de pior caso do algoritmo desenvolvido.

O algoritmo percorre o *array* e a cada comparação entre os elementos, verifica e, caso necessário, troca o maior elemento com o menor. Ao chegar na última posição do *array* e após realizar a verificação final, o menor elemento estará na primeira posição da lista. Depois, percorre o array, realiza as comparações e caso necessário faz as trocas. Ao alcançar o final desta rodada, o segundo menor valor estará na segunda posição do *array*. Este procedimento é realizado sucessivamente, até que a lista esteja toda classificada em ordem crescente.

4 (2.0). Relacione V (verdadeiro) ou F (falso) para cada alternativa abaixo. **Caso a alternativa seja falsa, justifique o erro. Não serão consideradas as alternativas sem justificativa ou com justificativa incorreta**.

( ) Os algoritmos de classificação *InsertionSort*, *MergeSort* e *QuickSort* tem uma eficiência melhor se comparados aos algoritmos *SelectionSort* e *BubbleSort*. Essa afirmação é válida, pois a complexidade de pior caso dos três primeiros é O(n log n) enquanto dos dois últimos é $O(n^2)$.

( ) Tanto o algoritmo *MergeSort* quanto o *QuickSort* utilizam a técnica chamada divisão e conquista para subdividir os elementos do array que será ordenado. A principal diferença entre eles é que o *MergeSort* concatena as soluções obtidas para chegar à solução final, enquanto que o *QuickSort* faz intercalações.

( ) A principal vantagem de uma lista estática (array) em relação à uma lista dinâmica é a possibilidade de inserção e remoção de um elemento em qualquer posição.

( ) Pilha é um tipo de lista linear que segue a política: o primeiro elemento que entra na lista é o último elemento que sai.

( ) Inicializando a variável x com o valor 24, a função *desafio* a seguir terá como resultado o valor 6. Para justificar a afirmativa, apresente o chinês.

```
int desafio (int x)
{
    if (x <= 12)
        return x * 2;
    else
        return desafio(desafio(x / 4));
}
```

Obs: A P1 vale 8.0 pontos. Os dois pontos restantes serão complementados pelas tarefas #1 e #2..