INSTITUTO FEDERAL
BRASÍLIA
Campus Taguatinga

AVALIAÇÃO DE APRENDIZAGEM

CURSO: ABI Ciência da Computação DATA: 04/05/2017
DISCIPLINA: Estrutura de Dados PROFESSOR: ROBERTO FONTES

ALUNO: _____

Orientações gerais para a realização da prova.

- Fazer a avaliação com **caneta azul ou preta**.
- Identificar todas as páginas com o seu nome.
- Ler todas as páginas, verificando se há algum erro de impressão ou omissão.
- Não é permitida a consulta de documentos, de colegas e telefones celulares;
- Não se esqueça de fazer a correção ortográfica/gramatical de suas respostas;
- **A prova é estritamente individual.** **Total prova: 10 pontos.**

1. [1,0 ponto] No que se refere a estruturas de dados é INCORRETO afirmar:

a) Numa fila dupla, os elementos podem ser inseridos e removidos de qualquer um dos extremos da fila.
b) Em qualquer situação é possível usar uma única fila dupla para representar duas filas simples.
c) A implementação de uma fila dupla normalmente é mais eficiente com uma lista duplamente encadeada que com uma encadeada simples.
d) Pela definição de fila, se os elementos são inseridos por um extremo da lista linear, eles só podem ser removidos pelo outro.
e) Numa lista singularmente encadeada, para acessar o último nodo é necessário partir do primeiro e ir seguindo os campos de ligação até chegar ao final da lista.

2. [3,0 pontos] Escreva uma função que compara duas filas (***info é do tipo float***). A função deve receber um ponteiro para cada fila e retornar 1 caso as duas forem iguais e 0 caso contrário. **Obs Escreva todas as funções auxiliares caso necessário**

Protótipo da função: **int compara (Fila* f1, Fila* f2)** ;

3. [3,0 pontos] Escreva uma função que remova ***info*** em uma Lista duplamente encadeada.

Protótipo da função: **Lista2* remove_info (Lista2* lst, int info);**

4. [3,0 pontos] Um sistema de navegação armazena o caminho entre dois pontos como uma lista encadeada de dados do tipo Logradouro, descrito a seguir, onde o campo nome contém o nome do logradouro e o campo **prox** aponta para o próximo elemento da lista.

```
struct logradouro {
    char nome[51];
    struct logradouro *prox;
};
typedef struct logradouro Logradouro;
```

Considere o tipo abstrato de dados Pilha, definido para armazenar ponteiros para cadeias de caracteres e que implementa as funções descritas na tabela a seguir:

Pilha* pilha_cria (void);	Retorna o ponteiro para uma nova pilha alocada dinamicamente.
char* pilha_pop (Pilha* p);	Retira um elemento do topo de uma pilha. O ponteiro da pilha é passado como parâmetro, e o retorno é o valor deste elemento.
void pilha_push (Pilha* p, char* x);	Insere um elemento no topo de uma pilha. O ponteiro da pilha e o elemento a ser inserido são passados como parâmetros.
int pilha_vazia (Pilha* p);	Verifica se a pilha está vazia. O ponteiro da pilha é passado como parâmetro e o retorno é 1, se a pilha está vazia, ou 0, caso contrário.
void pilha_libera (Pilha* p);	Esvazia e libera a memória alocada para uma pilha. O ponteiro da pilha é passado como parâmetro

Crie uma função em C para comparar dois caminhos e verificar se um é o inverso do outro, ou seja, comparar as duas listas que armazenam caminhos e verificar se o conteúdo de uma lista --- sequencia de nomes de logradouros armazenados em cada nó --- equivale ao da outra lista na ordem inversa. **Essa função deve usar o tipo abstrato Pilha para auxiliar nessa comparação**. Os parâmetros da função são os ponteiros l1 e l2 para as duas listas. O retorno da função deve ser 1 se uma lista for o inverso da outra e 0, caso contrário.

Protótipo da função: **int Compara_listas(Logradouro* l1, Logradouro* l2);**