DISCIPLINA DE ESTRUTURA DE DADOS
Trabalho 01 – 05/2018 (2018/1) – 03 pontos

Orientações:

- Escolha **2 estruturas de dados** e implemente os códigos solicitados;
- Organize seus códigos separando a interface da implementação;
- Entregue os códigos prontos em duas pastas compactadas através da tarefa disponibilizada no ambiente Moodle, impreterivelmente, até às 23h59min do dia 06/06/2018;
- Você deve apresentar os códigos funcionando em data estabelecida pelo professor no dia 06/06/2018.

1 - LISTA SIMPLESMENTE ENCADEADA (1,5)

Considere as seguintes definições:

```
typedef struct {
    int num;
    char nome[10];
} Contato;

typedef struct nodo Nodo;
```

```
struct nodo {
    Contato info;
    Nodo *prox;
};

typedef struct {
    Nodo *inicio;
} ListaSE;
```

```
#define SUCESSO 0
#define LISTA_VAZIA 1
#define FALTOU_MEMORIA 2
#define CONTATO_INEXISTENTE 3
```

Implemente as seguintes operações sobre uma lista simplesmente encadeada:

criaLista (0,1) **Saída**: uma lista. **Retorno**: nenhum. Descrição: cria uma lista vazia.	**consultaContato (0,3)** **Entrada**: uma lista e um número de telefone. **Saída:** um contato do tipo Contato. **Retorno**: SUCESSO, CONTATO_INEXISTENTE ou LISTA_VAZIA. Descrição: obtém os dados do contato que possui o número de telefone informado.
incluiAntes (0,3) **E/S:** uma lista. **Entrada:** um número de telefone e um contato do tipo **Contato.** **Retorno:** SUCESSO, CONTATO_INEXISTENTE ou FALTOU_MEMORIA. Descrição: inclui um contato antes do contato que possui o número de telefone informado.	**incluiDepois (0,3)** **E/S:** uma lista. **Entrada:** um número de telefone e um contato do tipo **Contato.** **Retorno:** SUCESSO, CONTATO_INEXISTENTE ou FALTOU_MEMORIA. Descrição: inclui um contato depois do contato que possui o número de telefone informado.
exibeLista (0,2) **Entrada**: uma lista. **Retorno**: nenhum. Descrição: exibe todos os nodos da lista.	**excluiContato (0,3)** **E/S**: uma lista. **Entrada:** um número de telefone**.** **Saída**: o contato excluído. **Retorno**: SUCESSO, CONTATO_INEXISTENTE ou LISTA_VAZIA. Descrição: exclui o contato que possui o número de telefone informado.

Considere que o programa cria uma LISTA SIMPLESMENTE ENCADEADA, executa a operação escolhida no cardápio de operações a seguir e exibe uma mensagem indicando se a operação foi ou não executada com sucesso.

Cardápio de Operações	***Descrição***
0. Fim	Encerra o programa.
1. Inclui contato antes	Lê o telefone e o nome de um contato e o inclui antes do número de telefone informado
2. Inclui contato depois	Lê o telefone e o nome de um contato e o inclui depois do número de telefone informado
3. Consulta contato	Imprime o nome do contato que possui o número de telefone informado
4. Exclui contato	Exclui da lista o contato que possui o número de telefone informado e exibe seu nome
5. Exibe lista	Exibe a lista de contatos

2 - PILHA SIMPLESMENTE ENCADEADA (1,5)

Considere as seguintes definições:

```
typedef struct {
    int cod;
    float salario;
} Funcionario;

typedef struct nodo Nodo;
```

```
struct nodo {
    Funcionario func;
    Nodo *prox;
};

typedef struct {
    Nodo *topo;
} PilhaSE;
```

```
#define SUCESSO 0
#define PILHA_VAZIA 1
#define FALTOU_MEMORIA 2
#define CODIGO_INEXISTENTE 3
```

Implemente as seguintes operações utilizando uma pilha simplesmente encadeada:

criaPilha (0,1) **Saída**: uma pilha vazia. **Retorno**: nenhum. Descrição: cria uma pilha vazia.	**exibePilha (0,3)** **Entrada**: uma pilha. **Retorno**: nenhum. Descrição: exibe todos os nodos da pilha.
estaVazia (0,2) **Entrada**: uma pilha. **Retorno**: **1** se a pilha está vazia e **0** caso contrário. Descrição: verifica se a pilha está vazia.	**empilha (0,3)** **E/S:** uma pilha. **Entrada:** um funcionário do tipo **Funcionario.** **Retorno:** SUCESSO ou FALTOU_MEMORIA. Descrição: inclui os dados de um funcionário no topo da pilha.
desempilha (0,3) **E/S**: uma pilha. **Saída**: os dados do funcionário armazenados no topo da pilha. **Retorno**: SUCESSO ou PILHA_VAZIA. Descrição: exclui os dados de um funcionário do topo da pilha.	**consultaExistencia (0,3)** **Entrada:** uma pilha e um código. **Retorno:** SUCESSO, PILHA_VAZIA ou CODIGO_INEXISTENTE. Descrição: verifica a existência de um funcionário que possui o código passado como argumento.

Considere que o programa cria uma PILHA SIMPLESMENTE ENCADEADA, executa a operação escolhida no cardápio de operações a seguir e exibe uma mensagem indicando se a operação foi ou não executada com sucesso.

Cardápio de Operações	*Descrição*
0. Fim	Encerra o programa.
1. Inclui um funcionário	Inclui um funcionário na estrutura de dados
2. Exclui um funcionário	Exclui um funcionário da estrutura de dados
3. Consulta a existência de um funcionário	Verifica a existência de um funcionário que possui o código passado como argumento
4. Exibe todos os funcionários	Exibe as informações de todos os funcionários cadastrados na estrutura de dados

3 - FILA SIMPLESMENTE ENCADEADA (1,5)

Considere as seguintes definições:

```
typedef struct {
    int num;
    char cia[10];
} Voo;

typedef struct nodo Nodo;
```

```
struct nodo {
    Voo v;
    Nodo *prox;
};

typedef struct {
    Nodo *frente;
    Nodo *ré;
} FilaSE;
```

```
#define SUCESSO 0
#define FILA_VAZIA 1
#define FALTOU_MEMORIA 2
#define NUMERO_INEXISTENTE 3
```

Implemente as seguintes operações utilizando uma pilha simplesmente encadeada:

criaFila (0,1) **Saída**: uma fila vazia. **Retorno**: nenhum. Descrição: cria uma fila vazia.	**exibeFila (0,2)** **Entrada**: uma fila. **Retorno**: nenhum. Descrição: exibe todos os nodos da fila.
quantidadeDeVoos (0,3) **Entrada**: uma fila. **Retorno**: a quantidade de aviões na fila. Descrição: verifica quantos aviões estão na fila para decolagem.	**insere (0,3)** **E/S:** uma fila. **Entrada:** um voo do tipo **Voo.** **Retorno:** SUCESSO ou FALTOU_MEMORIA. Descrição: inclui os dados de um voo no topo da fila.

retira (0,3)	**consultaExistencia (0,3)**
E/S: uma fila. **Saída**: os dados do voo que está na frente da fila. **Retorno**: SUCESSO ou FILA_VAZIA. Descrição: Retira um voo da fila.	**Entrada:** uma fila e um número. **Retorno:** SUCESSO, FILA_VAZIA ou NUMERO_INEXISTENTE. Descrição: verifica a existência de um voo que possui o número passado como argumento.

Considere que o programa cria uma FILA SIMPLESMENTE ENCADEADA, executa a operação escolhida no cardápio de operações a seguir e exibe uma mensagem indicando se a operação foi ou não executada com sucesso.

Cardápio de Operações	***Descrição***
0. Fim	Encerra o programa.
1. Insere um voo	Inclui um voo na estrutura de dados
2. Retira um voo	Exclui um voo da estrutura de dados
3. Consulta a existência de um voo	Verifica a existência de um voo que possui o número passado como argumento
4. Quantidade de aviões	Verifica a quantidade de aviões esperando na fila
5. Exibe todos os voos	Exibe as informações de todos os voos que estão na fila

BOM TRABALHO!