Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

Professora: Rúbia Eliza de Oliveira Schultz Ascari
Departamento Acadêmico de Informática (Dainf)
Tecnologia em Análise e Desenvolvimento de Sistemas
Estruturas de Dados 1

Exercícios sobre Listas Estáticas

1) Crie uma função que permite adicionar um item em uma posição qualquer da lista.

```
//adiciona um elemento na lista em posição qualquer
void adicionaItemLista(Lista *l, int chave, int pos) {
    if (verificaListaCheia(l)) {
        printf("Erro: a lista está cheia.\n");
        return;
    }
    if (pos > l->ultimo + 1) {
        printf("Erro: Posicao inválida.\n");
        return;
    }
    int nElementos = l->ultimo + 1;
    l->ultimo++;
    for (int i = nElementos; i > pos; i--) {
        l->item[i] = l->item[i-1];
    }
    Item novoItem;
    novoItem.chave = chave;
    l->item[pos] = novoItem;
}
```

2) Crie uma função que insere os itens em uma lista de forma ordenada.

```
//Adiciona item na lista mantendo a lista ordenada
void adicionaItemListaOrd(Lista *l, int chave) {
    int pos, i = 0;
    //Considerando que a lista já está ordenada, busca a posicao para
inserir o novo elemento
    while ((i <= l->ultimo) && (chave > l->item[i].chave))
        i++;
    return (adicionaItemLista(l, chave, i));
};
```

3) Altere a função que adiciona um item no fim da lista para que não seja permitido adicionar itens repetidos (itens com chaves/códigos iguais).

```
//adiciona um elemento no fim da lista
void adicionaItemFimLista(Lista *l, int chave) {
    if (verificaListaCheia(l)) {
        printf("Erro: a lista está cheia.\n");
        return;
    }
    for (int i = 0; i < l->ultimo+1; i++) {
        if (l->item[i].chave == chave) {
            printf("Erro: Nao e possivel inserir itens repetidos.\n");
            return;
        }
    }
```

```
    Item novoItem;
    novoItem.chave = chave;
    l->ultimo++;
    l->item[l->ultimo] = novoItem;
}
```

4) Crie uma função chamada copia_lista que recebe duas listas. A primeira deve ser uma lista vazia e a segunda não. A função deve copiar os itens da segunda lista na primeira lista.

```
void copiaListas(Lista *lDest, Lista *lOrig) {
    if (!verificaListaVazia(lDest)) {
        printf("Erro: A lista para a qual os itens serao copiados nao esta
vazia.\n");
        return;
    }
    for (int i = 0; i < lOrig->ultimo+1; i++) {
        adicionaItemFimLista(lDest, lOrig->item[i].chave);
    }
}
```

5) Crie uma função que recebe duas listas. Considere que dentro de uma lista não existem itens repetidos (você já tratou para que não seja permitido adicionar itens iguais). Sua função deve retornar uma terceira lista que representa a intersecção entre as duas listas recebidas.

```
Lista *intersecListas(Lista *lista1, Lista *lista2) {
    Lista *nova = criaListaVazia();
    for (int i = 0; i <= lista1->ultimo; i++)
    {
        for (int j = 0; j <= lista2->ultimo; j++) {
            if (lista1->items[i].chave == lista2->items[j].chave)
            adicionaItemFimLista(nova, lista1->items[i].chave);
        }
    }

    return nova;
}
```

6) Crie uma função que recebe duas listas. Considere que dentro de uma lista não existem itens repetidos. Sua função deve retornar uma terceira lista que representa a união entre as duas listas recebidas. Assegure-se de que a terceira lista não tenha itens repetidos.

```
//A função adicionaItensFimLista já impede a inclusão de itens repetidos.
Lista *uneListasSemRepeticao(Lista *lista1, Lista *lista2) {
    Lista *nova = criaListaVazia();
    copiaListas(nova, lista1);
    for (int i = 0; i <= lista2->ultimo; i++) {
        adicionaItemFimLista(nova, lista2->item[i].chave);
    }
    return nova;
}
```