Estruturas de Dados (COMP0405)

Exercícios 2 (Lista Sequencial Ordenada)

Professor: Alberto Costa Neto

Em todos os exercícios, deve-se partir da implementação da apresentada em sala de aula, que pode ser encontrada no seguinte endereço:
https://replit.com/@ALBERTOCOSTA3/ListaSequencialOrdenada

1) Implementar uma função para buscarEm um ***item*** na lista começando de um ponto inicial (***inicio***) e terminando no antecessor de ***fim***, isto é, desconsiderando as posições menores ***inicio*** e maiores ou iguais a ***fim***. Caso não encontre, a função retorna o valor -1. Aproveite e ajuste a função ***buscar*** padrão para utilizar a função ***buscarEm***, evitando assim duplicação de código. Lembre-se de que a lista está ordenada e você precisa utilizar a busca binária para ser mais eficiente. A função deve ter a assinatura abaixo:

```
int buscarEm(ITEM item, int inicio, int fim, LISTA *l);
```

2) Modificar a função ***inserir*** para encontrar a posição de inserção utilizando a busca binária. **Dica:** adapte a busca binária para retornar a posição de em que o item deveria ser inserido, já que a implementação padrão é retornar -1 quando o item não é encontrado.

3) Implementar uma função ***clonar*** para criar uma lista nova contendo uma cópia dos itens da lista passada como parâmetro. A função deve ter a assinatura abaixo:

```
LISTA * clonar(LISTA *l);
```

4) Implementar uma nova Lista Sequencial Ordenada que mantenha os itens em variáveis dinâmicas, ou seja, no array ***itens*** serão armazenados apenas os apontadores para os itens da lista, ao invés dos próprios itens. Lembre-se de alocar uma variável dinâmica para o item ao inserir e de desalocar a variável dinâmica quando fizer sua remoção ou limpar/destruir a lista. Aproveite e já ajuste as funções que implementou nas questões anteriores para esta implementação de Lista Sequencial.

5) Implementar uma nova Lista Sequencial Ordenada que mantenha os itens em um array alocado em uma variável dinâmica. A ideia é que o array vá crescendo gradativamente. Utilize como constantes o tamanho do bloco e o fator de crescimento percentual. Por exemplo, considerando uma lista com um bloco de 100 posições e um fator de crescimento 1 (100%), sempre que essa lista ficar completamente cheia, o vetor de itens deverá ser realocado para outra variável dinâmica capaz de armazenar 200 posições (porque o fator de crescimento é 1) e a inserção será feita sem problemas, passando a conter 101 itens e 99 posições vagas (2 blocos de 100). Quando chegar a 200 posições ocupadas, a próxima expansão será para 400 posições, ou seja, 100% sobre o tamanho anterior, já que o fator é de 1. Após ocorrerem várias remoções, é possível reduzir a utilização de memória da lista. Para isso, implemente uma função chamada ***compactar*** que recebe a lista, analisa quantos blocos precisa efetivamente e realoca a lista para um espaço menor. Lembre que o tamanho do bloco tem que ser respeitado. Por exemplo, se há 303 valores na lista e o tamanho do bloco for 100, será necessário um vetor de 400 posições. Lembre também que,

ao chamar a função ***limpar***, deve-se reduzir a capacidade da lista para o tamanho mínimo do bloco. Aproveite e já adeque as funções que já desenvolveu nas questões anteriores para esta implementação de Lista Sequencial Ordenada. **Dica:** Use a função ***realloc***.

```
void compactar(LISTA *l);
```