```
1: //   LISTA TADFILA
2: // jhonatan Figueiredo Almeida  20.1.8164
3:
4: /*1. Escreva uma função que verifica se duas listas simplesmente enc
5: int ListasIguais (Tlista L1, Tlista L2){
6:     if(L1.tamanho != L2.tamanho){
7:         return 0;
8:     }
9:     else{
10:        Tcelula *aux1 = L1.primeiro->prox;
11:        Tcelula *aux2 = L2.primeiro->prox;
12:
13:        while (aux1!=NULL){
14:            if(aux1->item.codigo != aux2->item.codigo){
15:                return 0;
16:            }
17:            aux1=aux1->prox;
18:            aux2= aux2->prox;
19:        }
20:    }
21:    return 1;
22:    // retorna 1 se forem iguais !!
23: }
24:
25: /*2. Construa uma função que concatena duas listas passadas como par
26:
27:
28: void JuntarLista (Tlista *L1, Tlista *L2){
29:    L1->ultimo->prox= L2->primeiro->prox;
30:    // Junta por endereço de memoria
31: }
32:
33: /*3. Construa uma função que recebe como parâmetro três listas L1, L
34: dividir a lista L1 em duas outras listas (dividir pela metade) que d
35: nas listas L2 e L3. Por exemplo, seja L1 = {A, B, C, D, E}, então, L
36: como: L2 = {A, B, C} e L3 = {D, E}*/
37:
38:
39: void Divide( Tlista L1, Tlista *L2 ,Tlista *L3){
40:
41:    Tcelula *aux;
42:    FLVazia (L2);
43:    FLVazia (L3);
44:
45:    int primeirametade= L1.tamanho/2;
46:
47:    aux = L1.primeiro->prox;
48:    for (int i=0; i<primeirametade ;i++){
49:        inserir(aux->item,L2);
```

```
50:         aux =aux->prox;
51:     }
52:     while (aux!=NULL){
53:         inserir(aux->item,L3);
54:         aux = aux->prox;
55:     }
56: }
57: /*4. Faça uma função que insere elementos em ordem crescente de nome
58: Para isso, pense nos três casos a seguir: (a) quando a lista estiver
59: utilize a função Insere padrão (que sempre insere um elemento no fin
60: em sala de aula, para inserir o elemento, veja que se ela está vazia
61: elemento sem se importar com a ordem da lista; (b) se o elemento a s
62: for maior do que o último elemento existente na lista, utilize a fun
63: se ele já é maior do que o último, significa que ele deverá entrar n
64: caso somente acontece quando os casos (a) e (b) não acontecem, assim
65: a lista comparando o elemento a ser inserido com os que já existem n
66: for maior do que o elemento comparado, continue percorrendo a lista,
67: caso ele seja menor ou igual, então você encontrou o ponto de inserç
68:
69: void IncrementaOrdem ( Tlista *L2){
70:
71:     Tproduto x;
72:     printf("Digite um item que deseja inserir em ordem !");
73:     Ler (&x);
74:     if(Vazia(*L2) || (strcmp(L2->ultimo->item.nome,x.nome) <=0) ){
75:         inserir(x,L2);
76:     }
77:     else{
78:         Tcelula *aux1 = L2->primeiro, *aux2,*aux3;
79:         while ((strcmp(aux1->prox->item.nome,x.nome))<=0){
80:             aux1=aux1->prox;
81:         }
82:         aux2=aux1->prox;
83:         aux3=(Tcelula*)malloc(sizeof(Tcelula));
84:         aux3->item=x;
85:         aux1->prox=aux3;
86:         aux3->prox= aux2;
87:     }
88:
89: }
90: /*5. Faça uma função que receba como parâmetro uma lista simplesment
91: os dados da i-ésima célula dessa lista. Tenha certeza de que a célul
92: se a lista for L1 = (A, B, C, D, E) e i = 3, você deverá realizar um
93: modo que avance apenas i vezes e consiga acessar a terceira célula e
94: C.*/
95:
96: void PesquisaCelula (Tlista *L, int j){
97:     // Poderia colocar para o usurio digitar O J aqui na função , ficari
98:     /*printf("\nDigite um numero da lista que desja ver :");
```

```
 99:     scanf("%d",&j);
100:     e não precisaria de passsar o j*/
101:
102:     Tcelula *aux1 = L->primeiro ->prox;
103:     int i=1;
104:     // Fiz IF caso o usuario seja "Inteligente" e digite numero que não
105:     if(j<=L->tamanho){
106:
107:         while (aux1 !=NULL){
108:             if(i==j){
109:                 printf("\nItem encontrado:");
110:                 ImprimirProduto(aux1->item);
111:
112:
113:             }
114:             aux1=aux1->prox;
115:             i++;
116:         }
117:     }
118:     else {
119:         printf("para de ser Trouxa, não tem esse item!!");
120:     }
121:
122:
123:
124: }
125:
126: /*6. Baseado na lógica da questão anterior, faça uma função que rece
127: lista simplesmente encadeada e remova a i-ésima célula dessa lista.*
128:
129:
130: void RemoverItem(Tlista *L1, int j){
131:     Tcelula *aux1 = L1->primeiro,*aux2,*aux3;
132:     int i=1;
133:     while (aux1->prox!=NULL && i<=j){
134:         if(i==j){
135:             aux2=aux1->prox;
136:             aux3->prox= aux2;
137:             break;// para o loop, não alocar endereço o aux3
138:         }
139:         aux3=aux1;
140:         aux1=aux1->prox;
141:         i++;
142:         free(aux1);
143:     }
144: }
145: // RemoveItem2 esta mais "simples"
146: void RemoverItem2(Tlista *L1, int j){
147:     Tcelula *aux1 = L1->primeiro,*aux2,*aux3;
```

```
148:     int i=1;
149:
150:     while (i<=j){
151:         aux3=aux1;
152:         aux1=aux1->prox;
153:         i++;
154:     }
155:         aux2=aux1->prox;
156:         aux3->prox= aux2;
157: }
158:
```