Universidade Federal do ABC

BC1424 - Algoritmos e Estruturas de Dados I
Gabarito da Prova P_1

1. Considere a estrutura da lista encadeada na Questão 4. Implemente um método que faz a busca sequencial recursiva em uma lista simplesmente encadeada `L`. Cabeçalho do método:
   `NoLista* buscaRecursivaComRetorno(Apontador pt, int chave)`

   O método deve retornar um ponteiro para o nó em que a chave se encontra ou `NULL` se a chave não foi encontrada.

   RESPOSTA:

```
NoLista* buscaRecursivaComRetorno(Apontador pt, int chave){
    if (pt!=NULL){
        if (pt->item.chave!=chave){
            buscaRecursivaComRetorno(chave,pt->prox);
        } else return pt;
    } else return NULL;
}
```

2. Implemente um método que faz a busca sequencial recursiva em uma lista estática `L` cuja estrutura é:

```
typedef struct {
    char nome[80];
    int chave;
} tipo_lista;
```

   A lista é assim declarada:
   `int n=50; tipo_lista L[50];`

   Cabeçalho do método:
   `int buscaSeqRec(int n, tipo_lista L[], int chave)`

   O método deve retornar o índice em que a chave se encontra ou `-1` se a chave não foi encontrada.

   RESPOSTA:

```
int buscaSeqRec(int n, tipo_lista L[], int chave) {
    if (n==0) {
        return -1;
    } else if (chave==L[n-1].chave) {
        return n-1;
    } else {
        return buscaSeqRec(n-1,L,chave);
    }
}
```

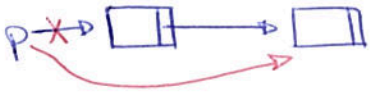
3. Assuma que p e q são dois nós em uma lista simplesmente encadeada cuja estrutura é descrita na Questão 4. Através de desenhos, explique o que acontece nas atribuições abaixo. Certifique-se de ilustrar o estado dos nós **ANTES** e **DEPOIS** da atribuição.

   (a) `p->item = q->item;`

   RESPOSTA:

   (b) `p = p->prox;`

   RESPOSTA:



   (c) `p = q;`

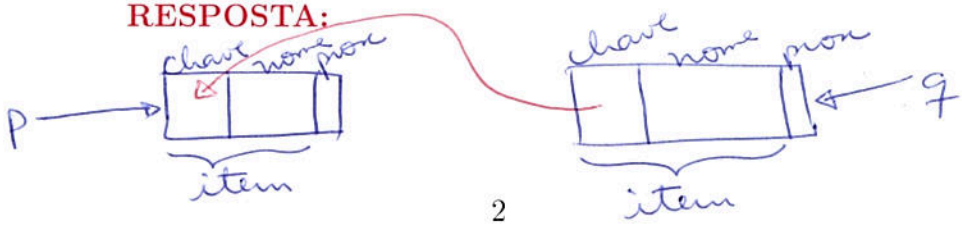
   RESPOSTA:

   (d) `p->prox = q;`

   RESPOSTA:

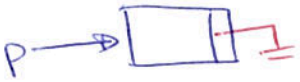
   (e) `p->item.chave = q->item.chave;`

   RESPOSTA:



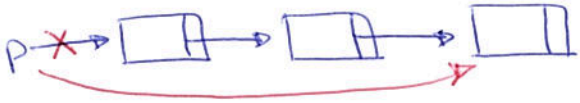
(f) `p->prox = NULL;`

RESPOSTA:



(g) `p=p->prox->prox;`

RESPOSTA:



(h) `p->prox = q->prox;`

RESPOSTA:

4. Dada uma lista simplesmente encadeada `L1` com a seguinte estrutura:

```
typedef struct {
    char nome[80];
    int chave;
} TipoItem;
typedef struct No *Apontador;
typedef struct No {
    TipoItem item;
    Apontador prox;
} NoLista;
```

Escreva um método para inverter a lista `L1` colocando o resultado na própria `L1`.

RESPOSTA:

```
void inverteLista(Apontador *L1){
    Apontador p=*L1,q=NULL,t;
    while (p!=NULL){
        t=p->prox;
        p->prox=q;
        q=p;
        p=t;
    }
    *L1=q;
}
```

5. Dada uma lista estática `L` cuja estrutura é:

```
typedef struct {
    char nome[80];
    int chave;
} tipo_lista;
```

A lista é declarada como segue:
`tipo_lista L[50];`
E assumindo que as chaves podem repetir, forneça as chaves que aparecem o menor (ou maior) número de vezes. A saída deve ser as chaves e o número de vezes que elas ocorrem.

RESPOSTA:

```
void contaChaves(tipo_lista L[], int n){ \\ n=número chaves
    int qtde[n];
    int chaves[n];
    int m=0;
    for(int i=0; i<n; i++){
        bool achou=false;
        for(int j=0; j<m; j++){
            if (chaves[j]==L[i].chave) {
                qtde[j]=qtde[j]+1;
                achou=true;
            }
        }
        if (!achou){
            chaves[m]=L[i].chave;
            qtde[m]=1;
            m++;
        }
    }
    // determinando menor (ou maior) ocorrência de chaves
    int menor=n;                    \\ int maior=0;
    for(int j=0; j<m; j++){
        if (qtde[j]<menor)          \\ if (qtde[j]>maior)
             menor = qtde[j];       \\ maior = qtde[j];
    }
    // imprimindo chaves com menor (ou maior) ocorrência
    for(int j=0; j<m; j++){
        if (qtde[j]==menor)         \\ if (qtde[j]==maior)
            cout << "Chave: " << chaves[j];
            cout << " Qtde: " << qtde[j] << endl;
    }
```

```
}
```

6. Implemente um método usando pilha estática para checar se a sequência de parênteses, colchetes e chaves de uma expressão matemática em uma string está correta. Por exemplo, a expressão (()[()]) está correta, mas a expressão (()())) não está.

RESPOSTA:

```
// retorna 1 se expressão ok, caso contrário retorna 0
int bemFormada(char s[], int n){
   char pilha[n];
   int t,i;
   t = 0;
   for (i = 0; i < n; ++i) {
      switch (s[i]) {
         case ')': if (t != 0 && pilha[t-1] == '(')
                      --t;
                   else return 0;
                   break;
         case ']': if (t != 0 && pilha[t-1] == '[')
                      --t;
                   else return 0;
                   break;
        case '(': pilha[t++] = s[i];
                   break;
        case '[': pilha[t++] = s[i];
                   break;
      }
   }
   return t == 0;
}
```