

Programação Imperativa, Teste, 16 de Maio de 2024

1. Adapte a função `fizzbuzz` definida abaixo por forma a devolver quantos números são impressos entre a primeira linha do tipo FizzBuzz e a primeira linha do tipo Buzz que se lhe segue? Se não aparecer nenhuma linha do tipo Buzz depois de uma linha do tipo FizzBuzz deve devolver -1.

```
int fizz(int n) {...}
int buzz(int n) {...}
int fizzbuzz(int n) {
    for (int i = 0; i < n; i++) {
        if (fizz(i) && buzz(i)) printf("FizzBuzz\n");
        else if (fizz(i))      printf("Fizz\n");
        else if (buzz(i))     printf("Buzz\n");
        else                  printf("%d\n", i);
    }
    return 0;
}
```

2. Implemente a função `void rodaEsq(int a[], int N, int r)` por forma a rodar os elementos de um array com N inteiros r posições para a esquerda (assuma que $r < N$). Por exemplo, se o array tiver o conteúdo {1,2,3,4,5,6} e se $r == 2$ então o array deverá ficar com os valores {3,4,5,6,1,2}. Tente minimizar o número de escritas no array.

3. Implemente de forma iterativa a função `int delete(int n, LInt *l)` que apaga o n-ésimo elemento de uma lista ligada (para $n == 0$ a função deverá remover o primeiro elemento). Se tal não for possível a função deverá devolver um código de erro.

```
typedef struct lint_nodo {
    int valor;
    struct lint_nodo *prox;
} *LInt;
```

4. Defina uma função `int maxCresc(LInt l)` que calcula o comprimento da maior sequência crescente de elementos consecutivos numa lista. Por exemplo, se a lista for [1, 2, 3, 2, 1, 4, 10, 12, 5, 4], a função deverá retornar 4. Serão desvalorizadas soluções que consultem cada nodo da lista mais do que uma vez.

5. Implemente uma função `ABin folhaEsq(ABin a)` que devolve **a folha** mais à esquerda de uma árvore (ou `NULL` se não tem nenhuma folha). Uma folha é um nodo em que **ambas** as sub-árvores são vazias.

```
typedef struct abin_nodo {
    int valor;
    struct abin_nodo *esq, *dir;
} *ABin;
```

6. Defina uma função `int parentesisOk(char exp[])` que, dada uma string onde está armazenada uma expressão aritmética com vários tipos de parêntesis, testa se os parêntesis estão corretos. Por exemplo, se a expressão for "3 + [2 - (3 - x)] + 4" a função deve retornar verdadeiro, enquanto que para a expressão "3 + [2 - (3 - x)) + 4" deve retornar falso. Considere 3 tipos de parêntesis: '(',')', '[,]', e '{,}'.