

# Teste de Programação Imperativa

LCC/MIEF/MIEI

1 Junho 2019

## Parte A

Considere as seguintes definições de tipos:

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

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

1. Defina uma função void `strrev (char s[])` que inverte uma string.
2. Defina uma função int `remRep (char x[])` que elimina de uma string todos os caracteres que se repetem sucessivamente deixando lá apenas uma cópia. A função deverá retornar o comprimento da string resultante. Assim, por exemplo, ao invocarmos a função com uma vector contendo "aaabaaabbbbaaa", o vector deve passar a conter a string "ababa" e retornar o valor 5.
3. Defina uma função void `merge (int r [], int a[], int b[], int na, int nb)` que, dados vectores ordenados a (com na elementos) e b (com nb elementos), preenche o vector r (com na+nb elementos) com os elementos de a e b ordenados.
4. Apresente uma definição da função int `deProcura (ABin a)` que testa se uma árvore é de procura.
5. Defina uma função int `pruneAB (ABin *a, int l)` que remove (libertando o espaço respectivo) todos os elementos da árvore \*a que estão a uma profundidade superior a l, retornando o número de elementos removidos. Assuma que a profundidade da raiz da árvore é 1, e por isso a invocação `pruneAB(&a,0)` corresponde a remover todos os elementos da árvore a.

## Parte B

Podem-se representar números num programa C directamente na sua representação decimal. Para isso considere o seguinte tipo de dados para representar números positivos como listas de dígitos decimais.

```
typedef struct digito {  
    unsigned char val;  
    struct digito *prox;  
} *Num;
```

No campo val de cada nó irá ser armazenado um único dígito (número entre 0 e 9), e convencionase que a lista vazia representa o número 0. É ainda conveniente estabelecer que a ordem com que os dígitos aparecem na lista está invertida (i.e. na cabeça da lista aparece o dígito menos significativo). A título de exemplo, o número 1234 será representado pela lista 4→3→2→1.

1. Defina as funções:

- (a) Num fromInt(unsigned int i), e
- (b) unsigned int toInt(Num n)

que convertem as listas de dígitos de e para inteiros do C. Por exemplo, fromInt(123) deverá retornar a lista 3→2→1.

2. Defina a função Num addNum(Num a, Num b) que codifique o "algoritmo de adição" estudado na escola primária para adicionar dois Nums (listas de dígitos). Note que deve garantir que cada dígito da lista resultante se encontra na gama pretendida (entre 0 e 9). Exemplo: sendo a = 7→8→9, e b = 6→7→8→9 (representam respectivamente os números 987 e 9876), o resultado deverá ser a lista 3→6→8→0→1 (que representa 10863).
3. Defina a função Num mulDig(unsigned char dig, Num a) que multiplica um dígito (um número entre 0 e 9) por uma lista de dígitos.
4. Recorrendo às funções definidas nas duas últimas alíneas, defina a função Num mulNum(Num a, Num b) que multiplica dois números representados por listas de dígitos. (obs: se a sua função fizer uso de memória de trabalho auxiliar, não se esqueça que deve ser libertada pela própria função).