

## Ministério da Educação Universidade Tecnológica Federal do Paraná Câmpus Pato Branco



\*\*(Peso 2,0)\*\*

# 4ªAvaliação (5/Julho) – AE22CP – 2017/1

Instruções para a avaliação:	<pre>prof.: Jean P. Martins(jeanmartins@utfpr.edu.br</pre>
<ul> <li><u>Leia</u> a prova com atenção e coloque o seu nome</li> <li>A avaliação é <u>individual</u>.</li> <li>A <u>interpretação</u> da prova faz parte da avaliação.</li> <li>É <u>PROIBIDA</u> a consulta à Internet.</li> </ul>	
O <u>peso</u> de cada exercício está indicado ao final de cada exercício está indicado esta exercício está indicado esta exercício está indicado esta exercício está indicado exercício está exercício está exercício está exercício está exercício exercício está exercício está exercício está exercício exercício está exercício e	lo enunciado.
•	https://www.hackerrank.com/ae22cp-20171-sub
NOME:	HACKERRANK-ID:
	conjunto (Set) pode ser feita utilizando-se <i>Tabelas Hash</i> . Considere olisões por ENCADEAMENTO SEPARADO. Uma estrutura Set pode
<pre>#define CAPACIDADE 1021 // Utilize e typedef struct {</pre>	
Lista* tabela_hash[CAPACIDADE] } Set;	];
1. TABELAS HASH	
Tabelas hash: conjuntos (interseção)	**(Peso 3,5)**
Implemente uma função para interseção de dois cor	ijuntos de inteiros.
<pre>Set* intersecao(Set* A, Set* B);</pre>	
Formato de entrada	
respectivos elementos, tanto para A, quanto para B.	o número de elementos a serem inseridos. Seguido pelos Exemplo:
24 56 2 158 66 174 51 192 51 58 66 3 197 164 1 8	18 73 41 62 89 75 192 32 169 192 106
195 41 45 101 191 11 102 4 <b>Restrições</b>	
<ul> <li>Elementos cujo hash colidam, devem ser ins</li> </ul>	eridos <b>no início da lista correspondente</b> .
Output Format	
O conjunto interseção deve conter todos elementos de A portanto a saída será: 41	que também estejam em B. No exemplo de entrada acima, C={41},
Há um espaço ' ' após cada número impro PILHAS	esso, inclusive no final

Implemente uma função que verifique se uma string da forma "xCy" representa um palíndromo.

```
int palindromo(char* A);
```

Pilhas: palíndromo (xCy)\_\_\_\_\_\_

# Formato de entrada

O programa receberá como entrada uma string contendo o caractere 'C', o qual delimita o fim da primeira parte da string.

A string é um palíndromo se sua primeira parte for igual ao inverso da segunda parte.

#### **Output Format (Imprima)**

SIM ou NÃO

#### LISTAS

(\*\*Escolha apenas uma das duas questões abaixo\*\*)

(1) Listas: merge\_

\*\*(Peso 2,5)\*\*

Dadas duas listas ordenadas A e B, produza uma terceira lista ordenada C, contendo todos valores contidos em A e B. A função deve utilizar a abordagem *merge*, e portanto passar por cada elemento das listas A e B apenas uma vez.

#### Formato de entrada

O programa receberá como entrada o tamanho da lista A e seus elementos. O mesmo ocorrendo para a lista B.

1 5 10 20 30 40 50 90 100 7 2 7 15 25 60 70 80

#### **Output Format (Imprima)**

A lista ordenada C

1 2 5 7 10 15 20 25 30 40 50 60 70 80 90 100

#### (2) Listas: strcmp\_

\*\*(Peso 2,5)\*\*

Dadas duas strings de entrada str\_a e str\_b, transformá-las em listas de caracteres.

```
Lista* A = strParaLista(str_a);
Lista* B = strParaLista(str_b);
```

Implementar uma função strcmplist que compara as duas listas, analogamente à função strcmp (em string.h)

```
int strcmplist(Lista* A, Lista* B);
```

### Formato de entrada

O programa receberá como entrada duas strings, uma por linha.

#### **Output Format (Imprima)**

strcmplist deve retornar um número negativo se A precede B, zero se A igual a B e maior que zero se A sucede B. Isso pode ser usado para comparar seus resultados com o de strcmp. Seu programa deve ter como saída, o seguinte:

Se as listas A e B são iguais, imprima

```
A igual a B
```

Senão, se as listas A e B diferem na posição de um caractere c<sub>1</sub> em A e c<sub>2</sub> em B

```
Se c_1 < c_2: A menor que B
se c_1 > c_2: A maior que B
```

Para o exemplo acima, a saída seria "A menor que B", pois A e B diferem nos caracteres  $c_1$  = 'a',  $c_2$  = 'c' e  $c_1$  <  $c_2$ .

99999999999999**a**aaaaaaa

 $999999999999999\mathbf{c} \texttt{alksdfacds1}$ 

TEÓRICA\_\_\_\_\_\_\*\*(Peso 2,0)\*\*

Em geral, lista encadadeadas são classificadas como simples, ou duplas. Nas listas simples, cada item da lista tem apenas um ponteiro para o próximo item, enquanto nas duplamente encadeadas há também um ponteiro para o item anterior. Portanto, uma lista simples só pode ser percorrida do início ao fim, enquanto uma duplamente encadeada também pode ser percorrida do fim ao início. Como essa característica pode ser utilizada para diminuir o tempo gasto na obtenção do item em uma dada posição *i* na lista duplamente encadeada?