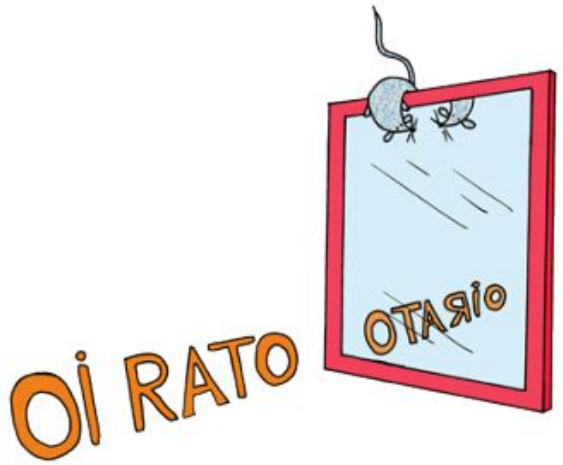
Manipulação de String

Vinicius e Eduardo

23/08/2018

Processamento de String

- Tópico frequente em Maratonas
- Identificar Palíndromos
- Identificar se uma palavra pertence a um dicionário



745262547

Classe String

- #include <string>
- Principais funções:
  - size = length
  - push_back
  - pop_back
  - compare
  - Operator+=
  - entre outras
- **Sempre consulte o c++ Reference!**

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Avance as Letras - URI 1607

É dado na entrada uma string A e outra B. Em uma operação você pode escolher uma letra da primeira string e avançar esta letra. Avançar uma letra significa transformá-la na próxima letra do alfabeto, veja que a próxima letra depois de z vem a letra a novamente!

Por exemplo, podemos transformar a string ab em bd em no mínimo 3 operações: ab -> bb -> bc -> bd. Podemos aplicar operações nas letras em qualquer ordem, outra possibilidade seria: ab -> ac -> bc -> bd.

Dadas as duas strings, calcule o mínimo número de operações necessárias para transformar a primeira na segunda.

Entrada

Na primeira linha terá um inteiro **T** (**T** ≤ 100) indicando o número de casos de teste.

Para cada caso, na única linha teremos as duas strings **A** ($1 \leq |\mathbf{A}| \leq 100$* ou $1 \leq |\mathbf{A}| \leq 10^{4}$** - sendo que |**A**| significa o tamanho da string **A**) e **B** (|**B**| = |**A**|* ou |**B**| = |**A**|**) separadas por um espaço. Ambas as strings são compostas por letras do alfabeto minúsculas apenas e são do mesmo tamanho.

*Ocorre em aproximadamente 90% dos casos de teste;

**Ocorre nos demais casos de teste.

Saída

Para cada caso imprima o número mínimo de operações.

Avance as Letras

Exemplo de Entrada	Exemplo de Saída
3 ab bd abc abc abcdefghiz aaaaaaaaaa	3 0 173

Uma Solução Possível

```cpp
#include <iostream>
#include <string>

using namespace std;

int countTo(char c1, char c2){
    int count = 0;
    c1 -= 97;
    c2 -= 97;

    for(; c1 != c2; c1 = (c1+1)%26)
        count++;

    return count;
}

int main(int argc, const char * argv[])
{
    int T;
    long long sum;
    string A, B;

    cin >> T;

    for(;T>0; T--){
        cin >> A >> B;
        sum = 0;
        for(int i = 0; i < A.length(); i++){
            sum += countTo(A[i], B[i]);
        }

        cout << sum << endl;
    }

    return 0;
}
```

Cifra de César - URI 1253

Júlio César usava um sistema de criptografia, agora conhecido como Cifra de César, que trocava cada letra pelo equivalente em duas posições à direita no alfabeto (por exemplo, 'A' vira 'C', 'R' vira 'T', etc.). Ao fim do alfabeto nós voltamos para o começo, isto é 'Y' vira 'A'. Nós podemos, é claro, tentar trocar as letras com quaisquer número de posições.

Entrada

A entrada contém vários casos de teste. A primeira linha de entrada contém um inteiro **N** que indica a quantidade de casos de teste. Cada caso de teste é composto por duas linhas. A primeira linha contém uma string com até 50 caracteres maiúsculos ('A'-'Z'), que é a sentença após ela ter sido codificada através desta Cifra de César modificada. A segunda linha contém um número que varia de 0 a 25 e que representa quantas posições cada letra foi deslocada para a direita.

Saída

Para cada caso de teste de entrada, imprima uma linha de saída com o texto decodificado (transformado novamente para o texto original) conforme as regras acima e o exemplo abaixo.

Cifra de César

Exemplo de Entrada	Exemplo de Saída
6 VQREQFGT 2 ABCDEFGHIJKLMNOPQRSTUVWXYZ 10 TOPCODER 0 ZWBGLZ 25 DBNPCBQ 1 LIPPSASVPH 4	TOPCODER QRSTUVWXYZABCDEFGHIJKLMNOP TOPCODER AXCHMA CAMOBAP HELLOWORLD

Uma Solução Possível

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    int movimentos, N;
    string str;
    cin >> N;
    for (int i = 0; i < N; i++)
    {
        cin >> str;
        cin >> movimentos;
        for (int j = 0; j < str.length(); j++){
            if ((str[j] - movimentos) < 'A')
                str[j] = (str[j] - movimentos) + 26;
            else
                str[j] = str[j] - movimentos;
        cout << str[j];
        }
        cout << endl;
    }
    return 0;
}
```