A3b. Взломщик!

Для хеширования строковых ключей, которые могут содержать строчные/прописные латинские буквы и цифры, используется следующая полиномиальная хеш-функция, значение которой определяется числовым параметром p:

```
size_t hash(std::string key) {
    const int p = ???;
    long long h = 0, p_pow = 1;
    for (size_t i = 0; i < s.length(); ++i) {
        h += (s[i] - 'a' + 1) * p_pow;
        p_pow *= p;
    }
    return h;
}
```

Для того, чтобы взломать хеш-функцию, требуется найти такой набор строк, который вызывает коллизии — одинаковые значения хеш-функции.

- (7 баллов) Одним из способов подбора строк, вызывающих коллизии, является поиск так называемых нейтральных элементов — строк, значение хеш-функции которых обращается в 0. Разработайте и обоснуйте алгоритм поиска строк, состоящих из двух символов, которые будут являться нейтральными элементами. Представьте обоснование и реализацию алгоритма. Ограничений на используемые языки программирования в этом задании нет.
- (3 балла) Найдите нейтральные элементы для всех значений параметра p ≤ 31.

1.

Будем считать что ***p*** натуральное число (иначе ограничение во втором пункте выглядело бы слишком страшным)

Для решения этой задачи нам понадобится таблица ascii

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

теперь мы можем понять что

```
(s[i] - 'a' + 1)
```

можно описать более удобно:

- если ***s[i]*** строчная буква, то это выражение будет равно ее номеру, начиная с единицы
- если ***s[i]*** прописная буква, то выражение будет равно ее номеру минус 32
- если же ***s[i]*** цифра, то выражение будет равно ей минус 67

тогда хеш-функция будет зануляться только если один из ее символов сточная буква, а другой либо прописная либо цифра, причем строчная всегда вторая.

в задании сказано найти алгоритм поиска всех строк из двух элементов подходящих под условие, и не сказано что это не должно быть брутфорсом, тем более в данном случае он работает за O(1) и вычисляет быстро. Тем не менее скорее всего авторы ожидают более элегантное решение, поэтому найду его

Вот такую функцию я написал:

Полный файл можно увидеть по ссылке: A3.cpp

```
std::vector<std::string> allnulls(int p){
    if(std::abs(p) < 2)
```

```
        return std::vector<std::string>();
    std::vector<std::string> ans;

    for(char i = '0'; i <= '9'; ++i){
        if(-26 * p <= (i - 'a' + 1) && (i - 'a' + 1) <= -1 * p && (i - 'a'
+ 1) % p == 0){
            std::string s;
            s+=i;
            s+='a' - (i - 'a' + 1)/p - 1;
            ans.push_back(s);
        }
    }
    for(char i = 'A'; i <= 'Z'; ++i){
        if(-26 * p <= (i - 'a' + 1) && (i - 'a' + 1) <= -1 * p && (i - 'a'
+ 1) % p == 0){
            std::string s;
            s+=i;
            s+='a' - (i - 'a' + 1)/p - 1;
            ans.push_back(s);
        }
    }

    return ans;
}
```

В отличии от бруфорса перебирает только первый символ

В ней я перебираю первый символ и проверяю существует ли подходящий второй. Вся арифметика взята из равенства

```
(s[0] - 'a' + 1) = (s[1] - 'a' + 1) * p;
```

Например для ***p = 5*** она дает результат

```
0x 2w 4v 6u 8t Bo Dn Fm Hl Jk Lj Ni Ph Rg Tf Ve Xd Zc
```

Проверим - действительно подходит.

Найдем все такие для каждого ***p***

Напишем вот такой код

A3.cpp

```
#include <iostream>
#include <vector>
#include <string>
```

```
std::vector<std::string> allnulls(int p){
    if(std::abs(p) < 2)
        return std::vector<std::string>();
    std::vector<std::string> ans;

    for(char i = '0'; i <= '9'; ++i){
        if(-26 * p <= (i - 'a' + 1) && (i - 'a' + 1) <= -1 * p && (i - 'a'
+ 1) % p == 0){
            std::string s;
            s+=i;
            s+='a' - (i - 'a' + 1)/p - 1;
            ans.push_back(s);
        }
    }
    for(char i = 'A'; i <= 'Z'; ++i){
        if(-26 * p <= (i - 'a' + 1) && (i - 'a' + 1) <= -1 * p && (i - 'a'
+ 1) % p == 0){
            std::string s;
            s+=i;
            s+='a' - (i - 'a' + 1)/p - 1;
            ans.push_back(s);
        }
    }

    return ans;
}

void coutvec(std::vector<std::string> &v){
    for(auto s : v){
        std::cout<<s<<' ';
    }
    std::cout<<'\n';
}

int main(){
    for(size_t p = 0; p <= 31; ++p){
        std::cout<<p<<" : ";
        std::vector<std::string> v = allnulls(p);
        coutvec(v);
    }
    return 0;
}
```

И получим вот такой вывод:

```
0 :
1 :
2 : 0x 2w 4v 6u 8t Bo Dn Fm Hl Jk Lj Ni Ph Rg Tf Ve Xd Zc
3 : 0p 3o 6n 9m Bj Ei Hh Kg Nf Qe Td Wc Zb
4 : 0l 4k 8j Dg Hf Le Pd Tc Xb
5 : 3i 8h Bf Ge Ld Qc Vb
```

```
6 : 0h 6g Be Hd Nc Tb Za
7 : 6f Dd Kc Rb Ya
8 : 0f 8e Hc Pb Xa
9 : 3e Ec Nb Wa
10 : 8d Bc Lb Va
11 : 4d Jb Ua
12 : 0d Hb Ta
13 : 9c Fb Sa
14 : 6c Db Ra
15 : 3c Bb Qa
16 : 0c Pa
17 : Oa
18 : Na
19 : Ma
20 : 8b La
21 : 6b Ka
22 : 4b Ja
23 : 2b Ia
24 : 0b Ha
25 : Ga
26 : Fa
27 : Ea
28 : Da
29 : Ca
30 : Ba
31 : Aa
```