128. Hash Function

* [Description](http://lintcode.com/en/problem/hash-function/" \l "description)
* [Notes](http://lintcode.com/en/problem/hash-function/#note)
* [Testcase](http://lintcode.com/en/problem/hash-function/#testcase)
* [Judge](http://lintcode.com/en/problem/hash-function/#judge)

In data structure Hash, hash function is used to convert a string(or any other type) into an integer smaller than hash size and bigger or equal to zero. The objective of designing a hash function is to "hash" the key as unreasonable as possible. A good hash function can avoid collision as less as possible. A widely used hash function algorithm is using a magic number 33, consider any string as a 33 based big integer like follow:

hashcode("abcd") = (ascii(a) \* 333 + ascii(b) \* 332+ ascii(c) \*33 + ascii(d)) % HASH\_SIZE

                              = (97\* 333 + 98 \* 332 + 99 \* 33 +100) % HASH\_SIZE

                              = 3595978 % HASH\_SIZE

here HASH\_SIZE is the capacity of the hash table (you can assume a hash table is like an array with index 0 ~ HASH\_SIZE-1).

Given a string as a key and the size of hash table, return the hash value of this key.f

<http://lintcode.com/en/problem/hash-function/#>

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package javaapplication1;

import java.util.\*;

public class JavaApplication1 {

public static int hashCode(char[] key, int HASH\_SIZE) {

// write your code here

long hash = 0, p = 1;

for (int i = key.length - 1; 0 <= i; --i) {

hash += (int)(key[i]) \* p;

hash %= HASH\_SIZE;

p = (p \* 33) % HASH\_SIZE;

}

return (int)(hash);

}

public static void main(String[] args) {

// TODO code application logic here

//key="abcd" and size=100,

char[] key = {'a','b','c','d'};

int HASH\_SIZE= 100;

System.out.println( hashCode( key, HASH\_SIZE));

}

}