**Maximum Occuring Character**

[string](http://practice.geeksforgeeks.org/tag-page.php?tag=string&isCmp=0)

Given a string, find the maximum occurring character in the string. If more than one character occurs maximum number of time then print the lexicographically smaller character.

**Input:**

The first line of input contains an integer T denoting the number of test cases. Each test case consist of a string in 'lowercase' only in a separate line.

**Output:**

Print the lexicographically smaller character which occurred the maximum time.

**Constraints:**

1 ≤ T ≤ 30

1 ≤ |s| ≤ 100

**Example:**

Input:  
2  
testsample  
geeksforgeeks

Output:  
e  
e

\*\*For More Examples Use Expected Output\*\*

<http://practice.geeksforgeeks.org/problem-page.php?level=-1&pid=87>

#include <iostream>

#include <stdio.h>

#include <map>

#include <string>

#include <algorithm>

#define ll long long int

#include <conio.h>

using namespace std;

int main() {

    int t;

    scanf("%d", &t);

    while(t--) {

        char  s[256];

        scanf("%s", s);

        fflush(stdin);

        //cout << s << endl;

        std::map<char, int> m;

        for(int i =0; i < strlen(s); i++) {

            m[s[i]]++;

        }

        char max\_char = s[0];

        int max\_frec = m[s[0]];

        for(std::map<char, int>::iterator it = m.begin(); it != m.end(); it++) {

            if(it->second > max\_frec) {

                max\_char = it->first;

                max\_frec = it->second;

            }

        }

        string letras = "";

        for(std::map<char, int>::iterator it = m.begin(); it != m.end(); it++) {

            if(it->second == max\_frec) {

               letras += it->first;

            }

        }

        std::sort(letras.begin(), letras.end());

        printf("%c**\n**", letras[0]);

    }

    getch();

    return 0;

}

**RAW Paste Data**