mkdir

工作中，每当要部署一台新机器的时候，就意味着有一堆目录需要创建。例如要创建目录“/usr/local/bin”，就需要此次创建“/usr”、“/usr/local”以及“/usr/local/bin”。好在，Linux下mkdir提供了强大的“-p”选项，只要一条命令“mkdir-p/usr/local/bin”就能自动创建需要的上级目录。现在给你一些需要创建的文件夹目录，请你帮忙生成相应的“mkdir-p”命令。

输入描述:

输入包含多组数据。

每组数据第一行为一个正整数 n(1≤n≤1024)。

紧接着 n 行，每行包含一个待创建的目录名，目录名仅由数字和字母组成，长度不超过 200 个字符。

输出描述:

对应每一组数据，输出相应的、按照字典顺序排序的“mkdir -p”命令。

每组数据之后输出一个空行作为分隔。

示例 1

输入

```
3
/a
/a/b
/a/b/c
3
/usr/local/bin
/usr/bin
/usr/local/share/bin
```

输出

```
mkdir -p /a/b/c

mkdir -p /usr/bin
```

```
mkdir -p /usr/local/bin
mkdir -p /usr/local/share/bin
```

```
// write your code here cpp
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;

int main()
{
    int n;
    while(cin>>n)
    {
        vector<string>v(n);
        vector<bool>flag(n,true);//标记
        for(int i=0;i<n;i++)
        {
            cin>>v[i];
        }
        sort(v.begin(),v.end());//有序化方便比对
        for(int i=0;i<n-1;i++)
        {
            //前后相同
            if(v[i]==v[i+1])
                flag[i]=false;
            //前串是后串的子串，而且后串后一位是 '/'
            elseif(v[i].size()<v[i+1].size()&&
v[i]==v[i+1].substr(0,v[i].size())&&v[i+1][v[i].size()]=='/')
                flag[i]=false;
        }
        for(int i=0;i<n;i++)
        {
            if(flag[i])
            {
                cout<<"mkdir -p "<<v[i]<<endl;
            }
        }
        cout<<endl;
    }
    return 0;
}
```