Problem: Bigger is Greater

Given a word , rearrange the letters of to construct another word in such a way that is lexicographically greater than . In case of multiple possible answers, find the lexicographically smallest one among them.

Input Format

The first line of input contains , the number of test cases. Each of the next lines contains .

Constraints

-
-
- will contain only lower-case English letters and its length will not exceed .

Output Format

For each testcase, output a string lexicographically bigger than in a separate line. In case of multiple possible answers, print the lexicographically smallest one, and if no answer exists, print `no answer`.

Sample Input 0

```
5
ab
bb
hefg
dhck
dkhc
```

Sample Output 0

```
ba
no answer
hegf
dhkc
hcdk
```

Explanation 0

- *Test case 1:*
  There exists only one string greater than `ab` which can be built by rearranging `ab`. That is `ba`.
- *Test case 2:*
  Not possible to rearrange `bb` and get a lexicographically greater string.
- *Test case 3:*
  `hegf` is the next string lexicographically greater than `hefg`.
- *Test case 4:*
  `dhkc` is the next string lexicographically greater than `dhck`.
- *Test case 5:*
  `hcdk` is the next string lexicographically greater than `dkhc`.

Solution

```
/*Sorts the string in increasing order after a particular character*/
string sort(string &str, int start)
{for(int i=start; i<str.length(); i++)
      {for(int j=start; j<str.length()-1; j++)
            {if(str[j]>str[j+1])
                  {char temp=str[j];
                        str[j]=str[j+1];
                        str[j+1]=temp;
                  }
            }
      }
return str;
}

/*Process the string for next permutatoin*/
string process(string str)
      {
            for(int i=str.length()-2; i>=0; i--)
                  {
                        for(int j=str.length()-1; j>i; j--)
                              {
                                    if(str[i]<str[j])
                                          {
                                                char temp=str[i];
                                                str[i]=str[j];
                                                str[j]=temp;
                                                sort(str,i+1);
                                                return str;
                                          }
                              }
                  }
      return str;
      }

int main()
      {
            string str;
            int cases;
            cin>>cases;
            for(int i=0; i<cases; i++)
             {
                cin>>str;
                (str==process(str) ? cout<<"no answer" :cout<<process(str));
                cout<<endl;
              }
      }                                                `‘ Anshul AgGarwal
```

Using STL

```
int main()
    {
        string str;
        int cases;
        cin>>cases;
        for(int i=0; i<cases; i++)
            {
                cin>>str;
                if( next_permutation( str.begin(),str.end() ) )
                        { cout<<str; }
                else
                        { cout<<"no answer"; }
                cout<<endl;
            }
    }
```

`’Anshul AgGarwal