F NeXT——by 张威

Description

没办法，名额有限，没有被选中还有下一次嘛。
于是，DPY 给了 Thor 一个候选序列，为一个不带有重复字母且只有大写字母的字符串。
他让 Thor 变换一些序列中的顺序使得新得到的这个字符串是比原来的字符串大的字符串中最小的一个。
Thor 做不出来，可是幸好还有你。

Input

多组数据。
每组数据仅有 1 行。为一个不带有重复字母且只有大写字母的字符串。(保证字符串中出现的字符都是连续的，若长度为 3，那么最大的字符为 C)

Output

对于每组数据输出仅 1 行。
输出一个字符串，为题目中所描述的。如果找不到这样的字符串请输出"What?"。

Sample Input

```
A
ACB
CBA
```

Sample Output

```
What?
BAC
What?
```

Hint

关于字符串的比较规则即：两个字符串自左向右逐个字符相比（按 ASCII 值大小相比较），直到出现不同的字符或遇'\0'为止。如：
"A"<"B" "a">"A" "computer">"compare"

解题分析

刚看到题，先是觉得简单，后来……不过还是想到了一个方法，在此感谢杨帆大神的提示。首先，我们需要将这个字符串读入，由于没给长度，只好就用字符串的知识来做，首先声明字符串（以下称为“串”），然后进行读入，while cin什么的。然后对串进行检查，这是由函数——judge 完成的，若长度为一或为倒序排列则输出“What？”，同时，函数在检查通过的情况下返回“断点值”，可作为弥补错误的工具，但是我最终没有用到。之后，我再次求出断点值（懒得引用了，总感觉不安全），该值在我的算法中是非常重要的，它表明了字符串从此值表示的位置（断点）开始向后都为递减字符。（递减含义请参照 Hint 理解）断点向前一个字符（以下称第一字符）之前的字符串显然是不应更改的，因为其大小不可减小，而如果增大，其变化量将必然大于仅改变第一个字符的变化量。因此，我们应将第一字符改变，改变应使此字符尽量少地增大，因而我在断点后寻找最小的比第一字符大的字符，然后将二者调换位置，最后将断点及以后的字符串按最小方式重排，即得所求字符串。

参考代码

```
#include<iostream>
#include<stdio.h>
#include<string>
#include<stdlib.h>//----------------------------------------------检错使用，请忽略。
using    namespace std;
void arrays(string    &,int ,int ,int );//---同样可忽略，想尝试按引用调用 string，出了些问题
int judge(string ,int );

int main()
{
        string A;
      while(cin>>A)
      {
       int n=A.length();
       if(judge(A,n)==0)//-----------------------------------------judge
       {cout<<"What?"<<endl;continue;}

  int leng=n, n1=n;
      while(n--)
      {
            if(A[n]>A[n-1])
                  break;
}//-----------------------------------------n 即为断点值。

                  char change;
                  int    min=n;//----------------------------A【min】为断点后符合条件的最小字符。
      while(n1>=n)
      {n1--;
            if(A[n1]>A[n-1]&&A[n1]<A[min])
                  min=n1;
      }change=A[min];
      A[min]=A[n-1];
      A[n-1]=change;
//-----------------------------------------------------------交换位置
      bool flag=false;
      leng--;
      while(flag==false)
      {int len=leng;flag=true;

        while(len>n)
      {
```

```
            if(A[len]<A[len-1])
            {change=A[len-1];
        A[len-1]=A[len];
        A[len]=change;
        flag=false;
            }len--;
        }
        }//------------------------------------------------------------冒泡法排序。
        cout<<A<<endl;
        }

}


int judge(string array,int len)
{
        bool flag=true;int loc;
        if(len==1)
            return 0;
        while (len--&&len>0)
        {
            if(array[len]>array[len-1])
                {flag=false;loc=len;
        }}
        if(flag==true)
return 0;
else
        return loc+1;

}//------------------------------------------------------------judge 的函数体。
```

做题果然重要，书可以不看，像这种题多做做，整本书都会了。

赠品：

String 类的函数：

```
char charAt(int index)
//返回指定索引处的 char 值。
int codePointAt(int index)
//返回指定索引处的字符（Unicode 代码点）。
int codePointBefore(int index)
//返回指定索引之前的字符（Unicode 代码点）。
int codePointCount(int beginIndex, int endIndex)
//返回此 String 的指定文本范围中的 Unicode 代码点数。
```

int compareTo(String anotherString)
//按字典顺序比较两个字符串。
int compareToIgnoreCase(String str)
//不考虑大小写，按字典顺序比较两个字符串。
String concat(String str)
//将指定字符串联到此字符串的结尾。
boolean contains(CharSequence s)
//当且仅当此字符串包含 char 值的指定序列时，才返回 true。
boolean contentEquals(CharSequence cs)
//当且仅当此 String 表示与指定序列相同的 char 值时，才返回 true。
boolean contentEquals(StringBuffer sb)
//当且仅当此 String 表示与指定的 StringBuffer 相同的字符序列时，才返回 true。
static String copyValueOf(char[] data)
//返回指定数组中表示该字符序列的字符串。
static String copyValueOf(char[] data, int offset, int count)
//返回指定数组中表示该字符序列的字符串。
boolean endsWith(String suffix)
//测试此字符串是否以指定的后缀结束。
boolean equals(Object anObject)
//比较此字符串与指定的对象。
boolean equalsIgnoreCase(String anotherString)
//将此 String 与另一个 String 进行比较，不考虑大小写。
static String format(Locale l, String format, Object... args)
//使用指定的语言环境、格式字符串和参数返回一个格式化字符串。
static String format(String format, Object... args)
//使用指定的格式字符串和参数返回一个格式化字符串。
byte[] getBytes()
//使用平台默认的字符集将此 String 解码为字节序列，并将结果存储到一个新的字节数组中。
void getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)
//已过时。 该方法无法将字符正确转换为字节。从 JDK 1.1 起，完成该转换的首选方法是通过 getBytes() 构造方法，该方法使用平台的默认字符集。
byte[] getBytes(String charsetName)
//使用指定的字符集将此 String 解码为字节序列，并将结果存储到一个新的字节数组中。
void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
//将字符从此字符串复制到目标字符数组。
int hashCode()
//返回此字符串的哈希码。
int indexOf(int ch)
//返回指定字符在此字符串中第一次出现处的索引。
int indexOf(int ch, int fromIndex)
//从指定的索引开始搜索，返回在此字符串中第一次出现指定字符处的索引。
int indexOf(String str)
//返回第一次出现的指定子字符串在此字符串中的索引。

//不包含该字符串返回 -1
int indexOf(String str, int fromIndex)
//从指定的索引处开始，返回第一次出现的指定子字符串在此字符串中的索引。
//不包含该字符串返回 -1
String intern()
//返回字符串对象的规范化表示形式。
int lastIndexOf(int ch)
//返回最后一次出现的指定字符在此字符串中的索引。
int lastIndexOf(int ch, int fromIndex)
//从指定的索引处开始进行后向搜索，返回最后一次出现的指定字符在此字符串中的索引。
int lastIndexOf(String str)
//返回在此字符串中最右边出现的指定子字符串的索引。
int lastIndexOf(String str, int fromIndex)
//从指定的索引处开始向后搜索，返回在此字符串中最后一次出现的指定子字符串的索引。
int length()
//返回此字符串的长度。
boolean matches(String regex)
//通知此字符串是否匹配给定的正则表达式。
int offsetByCodePoints(int index, int codePointOffset)
//返回此 String 中从给定的 index 处偏移 codePointOffset 个代码点的索引。
boolean regionMatches(boolean ignoreCase, int toffset, String other, int ooffset, int len)
//测试两个字符串区域是否相等。
boolean regionMatches(int toffset, String other, int ooffset, int len)
//测试两个字符串区域是否相等。
String replace(char oldChar, char newChar)
//返回一个新的字符串，它是通过用 newChar 替换此字符串中出现的所有 oldChar 而生成的。
String replace(CharSequence target, CharSequence replacement)
//使用指定的字面值替换序列替换此字符串匹配字面值目标序列的每个子字符串。
String replaceAll(String regex, String replacement)
//使用给定的 replacement 字符串替换此字符串匹配给定的正则表达式的每个子字符串。
String replaceFirst(String regex, String replacement)
//使用给定的 replacement 字符串替换此字符串匹配给定的正则表达式的第一个子字符串。
String[] split(String regex)
//根据给定的正则表达式的匹配来拆分此字符串。
String[] split(String regex, int limit)
//根据匹配给定的正则表达式来拆分此字符串。
boolean startsWith(String prefix)
//测试此字符串是否以指定的前缀开始。
boolean startsWith(String prefix, int toffset)
//测试此字符串是否以指定前缀开始，该前缀以指定索引开始。
charSequence subSequence(int beginIndex, int endIndex)
//返回一个新的字符序列，它是此序列的一个子序列。

String substring(int beginIndex)
//返回一个新的字符串，它是此字符串的一个子字符串。
String substring(int beginIndex, int endIndex)
//返回一个新字符串，它是此字符串的一个子字符串。
char[] toCharArray()
//将此字符串转换为一个新的字符数组。
String toLowerCase()
//使用默认语言环境的规则将此 String 中的所有字符都转换为小写。
String toLowerCase(Locale locale)
//使用给定 Locale 的规则将此 String 中的所有字符都转换为小写。
String toString()
//返回此对象本身（它已经是一个字符串！）。
String toUpperCase()
//使用默认语言环境的规则将此 String 中的所有字符都转换为大写。
String toUpperCase(Locale locale)
//使用给定的 Locale 规则将此 String 中的所有字符都转换为大写。
String trim()
//返回字符串的副本，忽略前导空白和尾部空白。
static String valueOf(boolean b)
//返回 boolean 参数的字符串表示形式。
static String valueOf(char c)
//返回 char 参数的字符串表示形式。
static String valueOf(char[] data)
//返回 char 数组参数的字符串表示形式。
static String valueOf(char[] data, int offset, int count)
//返回 char 数组参数的特定子数组的字符串表示形式。
static String valueOf(double d)
//返回 double 参数的字符串表示形式。
static String valueOf(float f)
//返回 float 参数的字符串表示形式。
static String valueOf(int i)
//返回 int 参数的字符串表示形式。
static String valueOf(long l)
//返回 long 参数的字符串表示形式。
static String valueOf(Object obj)
//返回 Object 参数的字符串表示形式