设想Caesar密码使用某个关键字的字母来决定文本中每个字母的位移。

例如：假设我们选择ace作为关键字。你可也以计算ace中的每个字母从字母a移位算起的位移量。这样a的位移量是0，c的位移量是2，e的位移量是4.

因此，当给定这个关键字ace后，文本的第一个字母的位移量是0，第二个字母的位移量是2，第三个是4，第四个是0，第五个位移量是2，依次类推。

例如：

ace： 024

加密前文本：this is a secret message

位      移：0240 24 0 240240 2402402

加密后文本：tjms jw a uictit oisuegg

写一个方法

String encode（String orginal,String key）

orginal是需要加密的字符串，key是密钥，返回加密后的字符串。

做一个解密方法

String decode(String text,String key)

text 需要解密的字符串，key是密钥，返回解密后的文本。

package p0630;

public class PasswordMaker {  
 public String encode(String orginal, String key) {  
  int cyc = key.length();  
  char[] keyChar = key.toLowerCase().toCharArray();  
  int[] off = new int[cyc];  
  for (int i = 0; i < cyc; i++) {  
   off[i] = keyChar[i] - 'a';  
  }  
  int len = orginal.length();  
  char[] orgChar = orginal.toCharArray();

  for (int i = 0, j = 0; i < len; i++) {  
   if (orgChar[i] == ' ') {  
    continue;  
   }  
   orgChar[i] += off[j % cyc];  
   j++;  
  }  
  return new String(orgChar);  
 }

 public String decode(String text, String key) {  
  int cyc = key.length();  
  char[] keyChar = key.toLowerCase().toCharArray();  
  int[] off = new int[cyc];  
  for (int i = 0; i < cyc; i++) {  
   off[i] = keyChar[i] - 'a';  
  }  
  int len = text.length();  
  char[] orgChar = text.toCharArray();

  for (int i = 0, j = 0; i < len; i++) {  
   if (orgChar[i] == ' ') {  
    continue;  
   }  
   orgChar[i] -= off[j % cyc];  
   j++;  
  }  
  return new String(orgChar);

 }

 public static void main(String[] args) {  
  PasswordMaker pm = new PasswordMaker();  
  String pass = pm.encode("this is a secret message", "ace");  
  String text = pm.decode(pass, "ace");  
  System.out.println(pass);  
  System.out.println(text);  
 }

}

#define C1 52845  
#define C2 22719  
#define C3 65  
  
//加密算法  
CString CEncryptDlg::Encrypt(CString S, WORD Key) // 加密函数  
{  
    CString Result,str;  
    int i,j;  
  
    Result=S; // 初始化结果字符串  
    for(i=0; i<S.GetLength(); i++) // 依次对字符串中各字符进行操作  
    {  
       Result.SetAt(i, S.GetAt(i)^(Key>>8)); // 将密钥移位后与字符异或  
       Key = ((BYTE)Result.GetAt(i)+Key)\*C1+C2; // 产生下一个密钥  
    }  
    S=Result; // 保存结果  
    Result.Empty(); // 清除结果  
    for(i=0; i<S.GetLength(); i++) // 对加密结果进行转换  
    {  
       j=(BYTE)S.GetAt(i); // 提取字符  
       // 将字符转换为两个字母保存  
       str="12"; // 设置str长度为2  
       str.SetAt(0, C3+j/26);//这里将65改大点的数例如256，密文就会变乱码，效果更好，相应的，解密处要改为相同的数  
       str.SetAt(1, C3+j%26);  
       Result += str;  
    }  
    return Result;  
}  
  
//解密算法  
CString CEncryptDlg::Decrypt(CString S, WORD Key) // 解密函数   
{   
    CString Result,str;   
    int i,j;   
    Result.Empty(); // 清除结果   
    for(i=0; i < S.GetLength()/2; i++) // 将字符串两个字母一组进行处理   
    {   
        j = ((BYTE)S.GetAt(2\*i)-C3)\*26;//相应的，解密处要改为相同的数   
        j += (BYTE)S.GetAt(2\*i+1)-C3;   
        str="1"; // 设置str长度为1   
        str.SetAt(0, j);   
        Result+=str; // 追加字符，还原字符串   
    }   
    S=Result; // 保存中间结果   
    for(i=0; i<S.GetLength(); i++) // 依次对字符串中各字符进行操作  
    {   
        Result.SetAt(i, (BYTE)S.GetAt(i)^(Key>>8)); // 将密钥移位后与字符异或   
        Key = ((BYTE)S.GetAt(i)+Key)\*C1+C2; // 产生下一个密钥   
    }   
    return Result;   
}   
  
//调用示例  
  
void CEncryptDlg::OnOK()   
{  
    // TODO: Add extra validation here  
    CString text=\_T("192.168.18.14");//需要加密的字符串   
    WORD key=1314;//key   
    CString jiami=Encrypt(text,key);//加密   
    AfxMessageBox(\_T("密文:")+jiami);   
    CString jiemi=Decrypt(jiami,key);//解密   
    AfxMessageBox(\_T("原文:")+jiemi);   
  
}