بگشای رمز

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

اخیراً مصطفی و پارسا از روشی کاملاً حرفه‌ای برای چت با یکدیگر استفاده می‌کنند. آن‌ها برای انتقال یک پیام، صرفاً ترتیب کلمات را تغییر می‌دهند. مصطفی و پارسا پس از انتقال **خود** کلمات پیام با ترتیب نامعلوم (که شیوه‌ی آن در این مسئله بررسی نمی‌شود)، **ترتیب و جایگشت آن‌ها** را تبدیل به یک رشته‌ی باینری می‌کنند. در حقیقت، رشته‌ی باینری حاوی اطلاعات فرایند مرتب‌سازی ادغامی بر روی جایگشت اولیه و اصلی کلمات است که در نهایت کلمات را به حالت مرتب‌شده در می‌آورد. گیرنده‌ی پیام باید با مهندسی معکوس، جایگشت اولیه‌ی کلمات که در حقیقت اصل پیغام است را بازیابی کند. نحوه‌ی تولید رشته در شبه‌کد زیر توضیح داده شده است:

```
mergeSort(a):
    b = a[0..(a.size / 2)]
    c = a[(a.size / 2)..a.size]
    mergeSort(b)
    mergeSort(c)
    a = merge(b, c)
```

```
merge(a, b):
    merge two sorted arrays a and b. in each step:
        if the element is chosen from a:
            add '0' to the binary string:
        else
            add '1' to the binary string
```

فرستنده در ابتدا تابع `mergeSort` را روی کل آرایه‌ی کلمات (که همان متن اصلی است) صدا زده و در نهایت با کد بالا، رشته‌ی باینری مورد نظر تولید و آماده‌ی ارسال می‌شود. کد رمزنگار، رشته‌ها را با شیوه‌ی خاصی مقایسه و نهایتاً مرتب می‌کند، به این‌صورت که اگر یکی از رشته ها شامل زیررشته‌ی `mosi` باشد، زودتر از رشته‌ی دیگر می‌آید. کوچکی و بزرگی حروف مهم نیست؛ `MosI` یا `mOSI` نیز همین‌طور هستند. اگر هر دو رشته این زیردنباله را داشته باشند یا هر دو نداشته باشند، رشته‌ها به‌صورت *lexicographically* مقایسه می‌شوند؛ به همان سیستمی که در لغت‌نامه‌ها لغت‌ها مرتب شده‌اند. اولویت حروف بزرگ را بالاتر در نظر بگیرید.

مصطفی و پارسا برای گسترش این شیوه‌ی رمزنگاری جهت استفاده در بین سایر دوستان‌شان، از شما می‌خواهند یک برنامه‌ی رمزگشایی برای‌شان بنویسید تا با دریافت یک پیام رمزنگاری‌شده، اصل متن را بازیابی کنید.

ورودی

در خط اول ورودی، عدد صحیح n وارد می‌شود که بیانگر تعداد رشته‌ها است.

$$1 \leq n \leq 10^5$$

در n خط بعدی، رشته‌ها با ترتیبی نامعلوم وارد می‌شوند. این رشته‌ها تنها شامل حروف کوچک و بزرگ انگلیسی هستند. طول هر رشته حداکثر ۵۰ کاراکتر است.

در خط بعدی، رشته‌ی باینری می‌آید.

خروجی

در n خط از خروجی، کلمات متن را با ترتیب اصلی‌شان چاپ کنید.

مثال

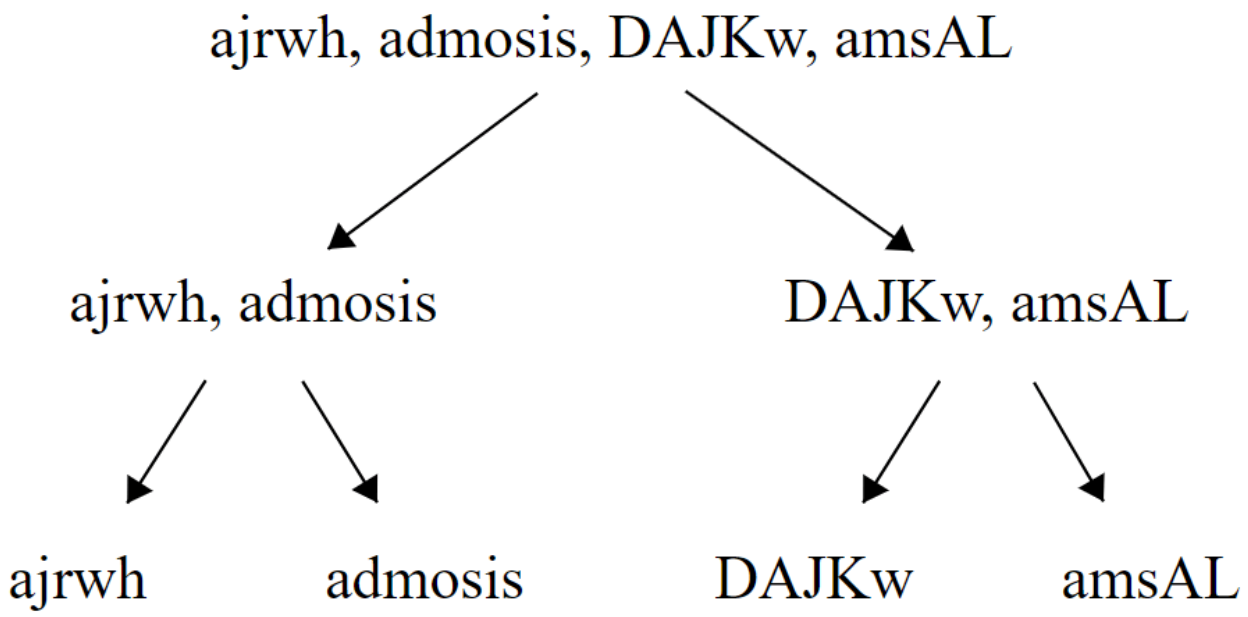
ورودی نمونه ۱

```
4
DAJKw
amsAL
ajrwh
admosis
10010101
```

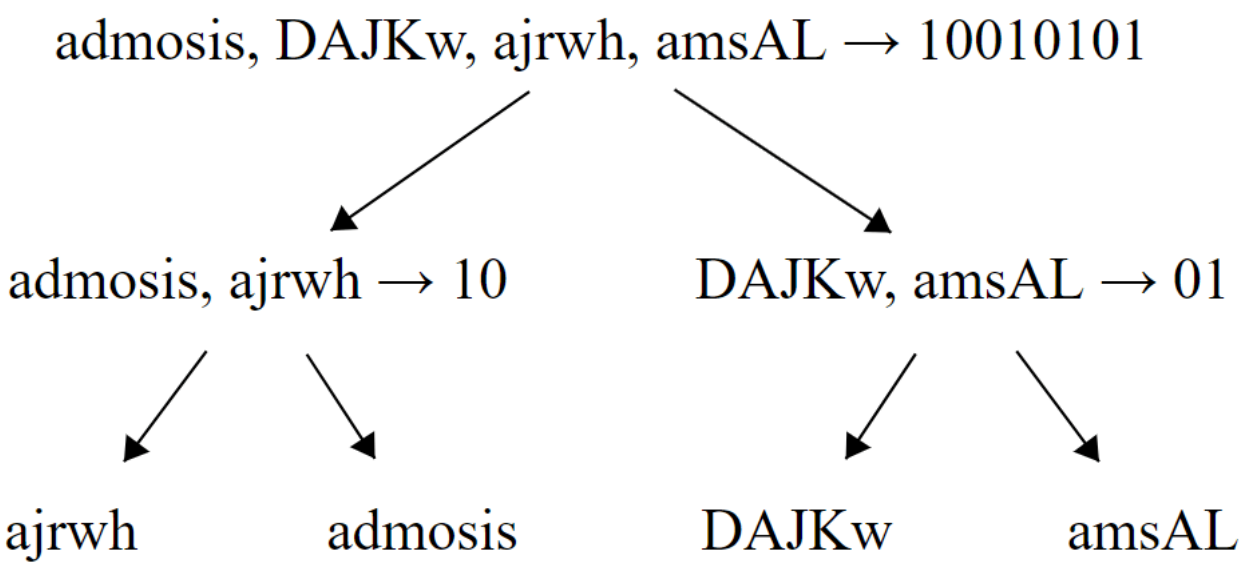
خروجی نمونه ۱

```
ajrwh
admosis
DAJKw
amsAL
```

درخت بازگشت مرتب‌سازی رشته‌ها با شروع از جایگشت اصلی به‌صورت زیر است:



ابتدا دو رشته‌ی `ajrwh` و `admosis` با یکدیگر مقایسه می‌شوند. از آن‌جایی که `admosis` شامل زیررشته‌ی `mosi` است، باید قبل از `ajrwh` بیاید. بنابراین، دو بیت ابتدایی رشته‌ی باینری برابر با `10` است. در ادامه، دو رشته‌ی `DAJKw` و `amsAL` با یکدیگر مقایسه می‌شوند. از آن‌جایی که هیچ‌یک از این دو رشته شامل زیررشته‌ی `mosi` نیستند، باید به‌صورت *lexicographically* مقایسه شوند. `D` اولویت بیش‌تری نسبت به `a` دارد، بنابراین `DAJKw` باید قبل از `amsAL` بیاید. پس، بیت‌های سوم و چهارم رشته‌ی باینری برابر با `01` هستند. در نهایت، باید عملیات ادغام را روی دو بخش `admosis,ajrwh` و `DAJKw,amsAL` انجام داد. ابتدا `admosis` می‌آید، پس بیت بعدی `0` است. سپس `DAJKw` می‌آید، پس بیت بعدی `1` است. در ادامه، `ajrwh` می‌آید، پس بیت بعدی `0` است. در نهایت `amsAL` می‌آید، پس آخرین بیت `1` است. رشته‌ی باینری به‌دست آمده، همان رشته‌ی باینری ورودی است.



ورودی نمونه ۲

```
7
ACSmosicLm
mOSICVKnHLYJs
NHCpGb
uRcGD
ZxMOsIC
cUUbXcf
RdEt
01110010101010010111
```

خروجی نمونه ۲

```
NHCpGb
ACSmosicLm
ZxMOsIC
mOSICVKnHLYJs
cUUbXcf
RdEt
uRcGD
```