رمزنگار

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

شرلوک پس از کشف راز قتل اندرسون توسط موری‌آرتی تصمیم می‌گیرد اسرار این جنایت مخوف را برای پلیس افشا کند به همین دلیل می‌خواهد نامه‌ای برای بازرس لستراد بنویسد و این راز را برملا کند. موری‌آرتی در شبکه پست مزدورانی دارد که نامه‌ها را به صورت مخفیانه بررسی می‌کنند؛ از آنجا که شرلوک نمی‌خواهد موری‌آرتی متوجه کشف این راز توسط او بشود، تصمیم می‌گیرد که نامه را به صورت رمزنگاری شده برای بازرس ارسال کند. در این سوال شما باید در رمزنگاری نامه به شرلوک کمک کنید!

نگاشت حروف به اعداد

ابتدا باید هر حرف در زبان انگلیسی را مستقل از بزرگ یا کوچک بودن، به یک عدد از 0 تا 25 نگاشت کنیم و در محاسبات رمزنگاری از آن عدد استفاده کنیم.

▼ **مشاهده جدول نگاشت**

حرف	عدد
A	0
B	1
C	2
D	3
E	4
F	5
G	6

حرف	عدد
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

الگوریتم‌های رمزنگاری

برای تبدیل متن انگلیسی معنادار (`plaintext`) به متن رمزشده (`ciphertext`) می‌توان از الگوریتم‌های مختلفی استفاده کرد:

▼ **الگوریتم `Additive Cipher`**

توضیح: در این روش، هر حرف متن اصلی با شیفت دادن به تعداد مشخصی از حروف در الفبا، به رمز تبدیل می‌شود.

پارامترها:

- `text` (متن اصلی)
- `key` (کلید، که یک عدد صحیح است و مقدار شیفت را مشخص می‌کند)

مثال:

- `text` : "hello"
- `key` : 3

رمزنگاری:

```
h -> K ((7 + 3) % 26 = 10 -> K)
e -> H ((4 + 3) % 26 = 7 -> H)
l -> O ((11 + 3) % 26 = 14 -> O)
l -> O ((11 + 3) % 26 = 14 -> O)
o -> R ((14 + 3) % 26 = 17 -> R)
```

نتیجه رمز شده: `"KHOOR"`

دستور:

```
additive-cipher -text "hello" -key 3
```

▼ **الگوریتم `Multiplicative Cipher`**

توضیح: در این روش، هر حرف متن اصلی با ضرب در یک کلید به رمز تبدیل می‌شود.

پارامترها:

- `text` (متن اصلی)
- `key` (کلید، که یک عدد صحیح است و با 26 نسبت به هم اول‌اند)

مثال:

- `text` : "hello"
- `key` : 3

رمزنگاری:

```
h -> V ((7 * 3) % 26 = 21 -> V)
e -> M ((4 * 3) % 26 = 12 -> M)
l -> H ((11 * 3) % 26 = 7 -> H)
l -> H ((11 * 3) % 26 = 7 -> H)
o -> Q ((14 * 3) % 26 = 16 -> Q)
```

نتیجه رمز شده: `"VMHHQ"`

دستور:

```
multiplicative-cipher -text "hello" -key 3
```

▼ **الگوریتم `Affine Cipher`**

توضیح: این روش ترکیبی از رمزنگاری جمعی و ضربی است، به این صورت که هر حرف متن اصلی ضرب در یک کلید می‌شود و سپس به نتیجه کلید دوم اضافه می‌شود.

پارامترها:

- `text` (متن اصلی)
- `a` (کلید ضربی، که یک عدد صحیح است و با 26 نسبت به هم اول‌اند)
- `b` (کلید جمعی، که یک عدد صحیح است)

مثال:

- `text` : "hello"
- `a` : 5
- `b` : 8

رمزنگاری:

```
h -> R ((7*5 + 8) % 26 = 43 % 26 = 17 -> R)
e -> C ((4*5 + 8) % 26 = 28 % 26 = 2 -> C)
l -> L ((11*5 + 8) % 26 = 63 % 26 = 11 -> L)
l -> L ((11*5 + 8) % 26 = 63 % 26 = 11 -> L)
o -> A ((14*5 + 8) % 26 = 78 % 26 = 0 -> A)`
```

نتیجه رمز شده: `"RCLLA"`

دستور:

```
affine-cipher -text "hello" -a 5 -b 8
```

▼ **الگوریتم `Mapping Cipher`**

توضیح: در این روش، هر حرف متن اصلی به یک حرف دیگر در الفبا مطابق با یک نگاشت از پیش تعریف شده تبدیل می‌شود.

پارامترها:

- `text` (متن اصلی)
- `mapping` (نگاشت حروف، به صورت رشته‌ای که ترتیب جدید حروف الفبا را نشان می‌دهد)

مثال:

- `text` : "hello"
- `mapping` : "zyxwvutsrqponmlkjihgfedcba" (نگاشت معکوس الفبا)

رمزنگاری:

```
h -> S
e -> V
l -> O
l -> O
o -> L
```

نتیجه رمز شده: `"SVOOL"`

دستور:

```
mapping-cipher -text "hello" -mapping "zyxwvutsrqponmlkjihgfedcba"
```

همان‌طور که در مثال‌ها آمده است، فرمت کلی هر دستور به شکل زیر است:

```
<cipher-type> -text "<text>" [-key <key>] [-a <a-value>] [-b <b-value>] [-mapp
```

که در آن `cipher-type` ، نوع الگوریتم رمز و سایر موارد، پارامترهای آن الگوریتم رمز هستند. پارامترهایی که بین `[]` آمده‌اند، بسته به الگوریتم رمزنگاری ممکن است در دستور وجود نداشته باشند.

نکات تکمیلی (مهم)

▼ نکته اول

ترتیب پارامترها می‌تواند متفاوت باشد. برای مثال دو دستور زیر معادل هستند:

```
affine-cipher -text "hello" -a 5 -b 8
affine-cipher -a 5 -text "hello" -b 8
```

▼ **نکته دوم**

فاصله‌های اطراف متن معنادار نادیده گرفته می‌شوند و در متن رمزشده مشاهده نمی‌شوند. اما فاصله‌های بین کلمات دقیقا در متن رمز شده می‌آید:

```
Plaintext: " please help me    "
Ciphertext: "qmfbtf ifmq nf"
```

▼ **نکته سوم**

بزرگی یا کوچکی حروف در متن معنادار مهم نیست. بنابراین متن رمزشده برای دو متن معنادار `hello` و `Hello` یکسان خواهد بود.

▼ **توجه: لازم است اصول کد تمیز تا حد ممکن در نظر گرفته شود.**

▼ **توجه: امکان پیاده‌سازی بخشی از سوال و کسب نمره آن بخش از سوال وجود دارد.**

▼ **توجه: تنها یک فایل شامل کدهای نوشته شده آپلود کنید.**

ورودی

ورودی شامل $n+1$ خط است. در خط اول n، تعداد دستورات می‌آید. در n خط بعدی، در هر خط یک دستور رمزنگاری آمده‌است.

$$1 \leq n \leq 50$$

خروجی

در خروجی به ازای هر دستور، نتیجه رمزنگاری را **با حروف بزرگ** در یک خط چاپ کنید.

مثال

ورودی نمونه ۱

```
2
additive-cipher -text "HELP me" -key 1
additive-cipher -text " HELP me " -key 1
```

خروجی نمونه ۱

```
IFMQ NF
IFMQ NF
```

▼ توضیحات نمونه ۱

در دستور اول، متن معنادار از نظر بزرگی یا کوچکی حروف فرقی نمی‌کند. بنابراین اگر `help me` را بخواهیم با الگوریتم `Additive Cipher` رمز کنیم، محاسبات به صورت زیر است:

```
h -> I ((7 + 1) % 26 = 8 -> I)
e -> F ((4 + 1) % 26 = 5 -> F)
l -> M ((11 + 1) % 26 = 12 -> M)
p -> Q ((15 + 1) % 26 = 16 -> Q)
m -> N ((12 + 1) % 26 = 13 -> N)
e -> F ((4 + 1) % 26 = 5 -> F)
```

- در این مثال، `key` برابر یک است. بنابراین هر کاراکتر در متن معنادار با عدد 1 جمع می‌شود. چون عدد حاصل باید بین 0 تا 25 باشد، باقیمانده‌ی حاصل جمع را بر عدد 26 بدست می‌آوریم. سپس در جدول نگاشت حروف انگلیسی به اعداد، حرف انگلیسی متناظر عدد بدست آمده را به عنوان کاراکتر رمزشده

درنظر می‌گیریم. کاراکتر فاصله بین کلمات نیز دقیقا در متن رمز شده می‌آید. بنابراین متن رمز شده، `IFMQ NF` خواهد بود.

- دستور دوم مشابه دستور اول است، با این تفاوت که فاصله‌های اضافی اطراف متن معنادار نادیده گرفته شده‌است. بنابراین متن رمزشده در هر دو دستور یکسان خواهد بود.

ورودی نمونه ۲

```
2
multiplicative-cipher -text "danger" -key 3
multiplicative-cipher -key 3 -text "danger"
```

خروجی نمونه ۲

```
JANSMZ
JANSMZ
```

▼ توضیحات نمونه ۲

- در دستور اول، اگر `danger` را بخواهیم با الگوریتم `Multiplicative Cipher` و کلید با مقدار 3 رمز کنیم، محاسبات به صورت زیر است:

```
d -> J ((3 * 3) % 26 = 9 -> J)
a -> A ((0 * 3) % 26 = 0 -> A)
n -> N ((13 * 3) % 26 = 13 -> N)
g -> S ((6 * 3) % 26 = 18 -> S)
e -> M ((4 * 3) % 26 = 12 -> M)
r -> Z ((17 * 3) % 26 = 25 -> Z)
```

در دستور دوم، صرفا جای پارامترها عوض شده و محاسبات مانند دستور قبل است.

ورودی نمونه ۳

```
1
affine-cipher -text "Hi" -a 3 -b 1
```

خروجی نمونه ۳

```
WZ
```

▼ توضیحات نمونه ۳

در این مثال، محاسبات رمزنگاری به صورت زیر است:

```
h -> W ((7*3 + 1) % 26 = 22 -> W)
i -> Z ((8*3 + 1) % 26 = 25 -> Z)
```

در این مثال، پارامتر `a` کلید ضربی و پارامتر `b` کلید جمعی است. بنابراین حروف متن معنادار در `a` ضرب و با `b` جمع می‌شوند. بنابراین متن رمز شده، `WZ` خواهد بود.

ورودی نمونه ۴

```
1
mapping-cipher -text "hello" -mapping "zyxwvutsrqponmlkjihgfedcba"
```

خروجی نمونه ۴

```
SVOOL
```

▼ توضیحات نمونه ۴

در این مثال، `mapping` داده شده برابر `"zyxwvutsrqponmlkjihgfedcba"` است. به عبارت دیگر نگاشت حروف انگلیسی به متن رمز شده به صورت زیر است:

```
a -> Z
b -> Y
c -> X
d -> W
e -> V
f -> U
g -> T
h -> S
i -> R
j -> Q
k -> P
l -> O
m -> N
n -> M
o -> L
p -> K
q -> J
r -> I
s -> H
t -> G
u -> F
v -> E
w -> D
x -> C
y -> B
z -> A
```

با توجه به نگاشت بالا، متن `hello` به این صورت رمز می‌شود:

```
h -> S
e -> V
l -> O
l -> O
o -> L
```

بنابراین متن رمز شده، `SVOOL` خواهد بود.

▼ الگوریتم ؟

۱. پردازش ورودی (Input Processing)

- خط اول ورودی: تعداد دستورات `n` .
- خطوط بعدی: هر خط یک دستور رمزنگاری است.
- هر دستور شامل موارد زیر است:
  - **نوع الگوریتم رمزنگاری:** مانند `additive-cipher` یا `affine-cipher` .
  - **پارامترها:** مانند `-text "hello" -key 3` . ترتیب پارامترها مهم نیست.

گام‌های اصلی:

۱. دریافت تعداد دستورات.

۲. دریافت هر دستور و جدا کردن نوع الگوریتم و پارامترهایش.

۳. اجرای الگوریتم مناسب بر اساس دستور.

۲. تبدیل حروف به عدد و بالعکس (Mapping Characters to Numbers)

- حروف انگلیسی (a تا z) به اعداد ۰ تا ۲۵ نگاشت می‌شوند:
  - `a → 0, b → 1, ..., z → 25`
- برای تبدیل عدد به حرف، از رابطه زیر استفاده می‌کنیم:
  - **عدد** → **(عدد % 26) + 'A'** (برای چاپ حروف بزرگ)

۳. الگوریتم‌های رمزنگاری (Cipher Algorithms)

الف) Additive Cipher (رمزنگاری جمعی)

- **فرمول:** $C=(P+K) \mod 26$
  - P: مقدار عددی حرف اصلی
  - K: کلید (key)
  - C: مقدار عددی حرف رمز شده

- **مثال:**
  - متن: `hello`
  - کلید: `3`
  - تبدیل: `h(7) → (7+3)%26 = 10 → K`

ب) Multiplicative Cipher (رمزنگاری ضربی)

- **فرمول:** $C=(P\times K) \bmod 26$
  - K باید با ۲۶ نسبت به هم اول باشد.
- **مثال:**
  - متن: `hello`
  - کلید: `3`
  - تبدیل: `h(7) → (7×3)%26 = 21 → V`

ج) Affine Cipher (رمزنگاری آفین - ترکیبی از جمعی و ضربی)

- **فرمول:** $C=(P\times A+B) \bmod 26$
  - A و B به ترتیب کلید ضربی و کلید جمعی هستند.
  - شرط: A باید با ۲۶ نسبت به هم اول باشد.
- **مثال:**
  - متن: `hello`
  - A=5, B=8
  - تبدیل: `h(7) → (7×5 + 8)%26 = 17 → R`

د) Mapping Cipher (رمزنگاری نگاشتی)

- **روش:**
  - نگاشت جدیدی برای الفبا تعریف می‌شود (مانند معکوس الفبا).
  - هر حرف اصلی به حرف متناظر در نگاشت جدید تبدیل می‌شود.
- **مثال:**

- متن: `hello`
- نگاشت: `"zyxwvutsrqponmlkjihgfedcba"`
- تبدیل: `h → S, e → V, l → O, o → L`

۴. تولید خروجی (Output Generation)

- متن نهایی باید:
  - **فقط با حروف بزرگ باشد.**
  - **فاصله‌های بین کلمات حفظ شوند.**
  - **فاصله‌های اضافه در ابتدا و انتهای متن حذف شوند.**
- در نهایت، هر دستور در یک خط جداگانه چاپ می‌شود.

▼ توابع

```
def process_commands(commands):
# این تابع لیستی از دستورات رو دریافت می‌کنه.
:# برای هر دستور
.1 # نوع الگوریتم و پارامترهای لازم رو با استفاده از parse_command جدا کن
 (additive, multiplicative, affine, mapping) .2 # بر اساس نوع الگوریتم
.#     تابع مربوطه رو فراخوانی کن و پارامترها رو بهش بده
.3 # نتیجه‌ی رمزنگاری شده رو چاپ کن
    pass


def parse_command(command):
 # این تابع یک دستور رشته‌ای مثل:
    # "additive-cipher -text 'hello' -key 3"
.# رو دریافت می‌کنه
.1 # ابتدا نوع الگوریتم رو از ابتدای دستور استخراج کن
.text، -key، -a، -b، -mapping- .2 # پارامترهای همراه مثل رو پیدا کن
.3 # مقادیر پارامترها رو در یک دیکشنری ذخیره کن و به همراه نوع الگوریتم برگردون
    pass
```

```
def char_to_num(char):
    # این تابع یک حرف انگلیسی (کوچک یا بزرگ) رو دریافت می‌کنه.
    # 1. از تابع ord() برای تبدیل حرف به مقدار عددی استفاده کن.
    # 2. مقدار یونیکد 'a' رو ازش کم کن تا عددی بین 0 تا 25 به دست بیاد.
    pass


def num_to_char(num):
    # این تابع یک عدد بین 0 تا 25 رو می‌گیره.
    # 1. از تابع chr() برای تبدیل عدد به کاراکتر استفاده کن.
    # 2. برای تبدیل به حروف بزرگ از یونیکد 'A' استفاده کن.
    # 3. اطمینان حاصل کن که عدد در بازه 0 تا 25 باقی بمونه (mod 26).
    pass


def additive_cipher(text, key):
    # این تابع متن اصلی و کلید عددی رو می‌گیره.
    # 1. روی هر حرف متن اصلی حلقه بزن.
    # 2. هر حرف رو به عدد تبدیل کن (با char_to_num).
    # 3. عدد به دست اومده رو با کلید جمع کن و حاصل رو mod 26 بگیر.
    # 4. عدد جدید رو به حرف تبدیل کن (با num_to_char).
    # 5. فاصله‌ها و کاراکترهای غیر الفبایی رو تغییر نده.
    pass


def multiplicative_cipher(text, key):
    # این تابع متن اصلی و کلید ضربی رو می‌گیره.
    # 1. روی هر حرف متن اصلی حلقه بزن.
    # 2. هر حرف رو به عدد تبدیل کن.
    # 3. عدد به دست اومده رو در کلید ضرب کن و حاصل رو mod 26 بگیر.
    # 4. عدد جدید رو به حرف تبدیل کن.
    # 5. قبل از شروع، بررسی کن که کلید نسبت به 26 اول باشه.
    pass


def affine_cipher(text, a, b):
    # این تابع متن اصلی و دو کلید ( a ضربی( و ) b جمعی( رو می‌گیره.
    # 1. روی هر حرف متن اصلی حلقه بزن.
    # 2. هر حرف رو به عدد تبدیل کن.
```

```
    # 3. از فرمول (a * num + b) % 26 برای رمزنگاری استفاده کن.
    # 4. عدد جدید رو به حرف تبدیل کن.
    # 5. مطمئن شو که a و 26 نسبت به هم اول هستند.
    pass


def mapping_cipher(text, mapping):
    # این تابع متن اصلی و یک رشته‌ی نگاشت (mapping) شامل 26 حرف رو می‌گیره.
    # 1. روی هر حرف متن اصلی حلقه بزن.
    # 2. هر حرف رو به عدد بین 0 تا 25 تبدیل کن.
    # 3. از عدد به عنوان ایندکس برای انتخاب حرف جدید از رشته mapping استفاده کن.
    # 4. حروف جدید رو به صورت بزرگ در نتیجه قرار بده.
    pass


def main():
    # این تابع ورودی رو از کاربر یا فایل می‌گیره.
    # 1. ورودی رو خط به خط بخون.
    # 2. خطوط ورودی رو به تابع process_commands بده.
    # 3. اجرای برنامه رو از اینجا مدیریت کن.
    pass
```

موفق باشید !