طرح ترافیک

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

می‌خواهیم یک سیستم برای طرح زوج و فرد پیاده‌سازی کنیم. در این سیستم هر نفر ممکن است چند ماشین داشته باشد. هر ماشین یک پلاک دارد که زوج یا فرد است. زوج یا فرد بودن یک پلاک بستگی به زوج یا فرد بودن رقم سمت راست پلاک دارد.

هر ماشین، می‌تواند در روزهای زوج یا فرد (بسته به پلاکش) در محدوده طرح زوج و فرد تردد کند و هیچ هزینه‌ای لازم نیست بدهد. ولی اگه در روز غیر از زوجیت پلاکش وارد طرح شود، دو حالت دارد، یا طرح برای آن روز نخریده و جریمه می‌شود یا طرح خریده که هیچ مشکلی نیست.

با خرید یک طرح d روزه برای یک خودرو، از **فردا**ی روزی که طرح خریداری شده است به مدت d روز، رفت آمد در طرح مشکلی ندارد.

در این سیستم، هر نفر باید یک حساب داشته باشد که در ابتدا اعتبار حسابش صفر است و می‌تواند اعتبار این حساب را افزایش دهد و یا برای یک ماشینش یک طرح بخرد. (توجه کنید حساب‌ها برای اشخاص است، نه ماشین‌ها).

توجه کنید جریمه شدن یک شخص برای ورود به طرح، از اعتبار حساب این شخص برای خرید طرح جداست. یعنی نباید مبلغ جریمه‌ها از حساب راننده‌ها کسر شود.

درخواست‌ها

▼ **درخواست** `REGISTER`

```
REGISTER <USERNAME> <TIMESTAMP>
```

در این درخواست یک نفر با نام <USERNAME> در لحظه <TIMESTAMP> به سیستم طرح ترافیک اضافه می‌شود.

- <USERNAME> یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد. تضمین می‌شود این زمان قابل قبول است.

در صورتی این فرد با موفقیت به سیستم اضافه می‌شود که قبلاً هیچ کسی با این <USERNAME> در سیستم ثبت نام نکرده باشد.

بعد از دریافت این درخواست، در صورتی که این فرد با موفقیت به سیستم اضافه شد پیام REGISTER DONE و در غیر این صورت پیام INVALID USERNAME را چاپ کنید.

مثالی از این درخواست:

```
REGISTER alireza 1402/10/15
```

▼ **درخواست REGISTER_CAR**

```
REGISTER_CAR <USERNAME> <CAR_PLATE> <TIMESTAMP>
```

در این درخواست یک نفر با نام <USERNAME> با شماره پلاک <CAR_PLATE> در لحظه <TIMESTAMP> به سیستم طرح ترافیک اضافه می‌کند.

- <USERNAME> یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- <CAR_PLATE> یک رشته به طول دقیقاً ۱۰ از ارقام است. این رشته می‌تواند با صفر آغاز شود.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

در صورت رخ دادن هر کدام از حالت‌های زیر دقیقا یکی از پیام‌های زیر را به الویت از بالا به پایین اجرا کنید:

- در صورتی که فردی با نام <USERNAME> وجود ندارد پیام INVALID USERNAME را چاپ کنید.

- در صورتی که این شماره پلاک (<CAR_PLATE>) برای یک خودرو دیگر ثبت شده است (حتی اگر خودرو برای همین راننده باشد) پیام INVALID CAR PLATE را چاپ کنید.
- در صورتی که هیچ کدام از اتفاقات بالا نیفتاد پیام REGISTER CAR DONE را چاپ کنید.

مثالی از این درخواست:

```
REGISTER_CAR alireza 7741181522 1402/10/15
```

▼ **درخواست NEW_RECORD**

```
NEW_RECORD <CAR_PLATE> <TIMESTAMP>
```

در این درخواست یک ماشین با شماره پلاک <CAR_PLATE> در لحظه <TIMESTAMP> وارد محدوده طرح ترافیک شده است.

- <CAR_PLATE> یک رشته به طول دقیقاً ۱۰ از ارقام است. این رشته می‌تواند با صفر آغاز شود.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

اگر ماشینی با این شماره پلاک در سیستم ثبت نشده است پیام INVALID CAR PLATE را چاپ کنید و در غیر این صورت اگر این درخواست در روز فرد آمده ولی پلاک ماشین زوج است یا درخواست در روز زوج آمده ولی پلاک ماشین فرد است و این ماشین برای این روز طرح ندارد پیام PENALTY RECORDED را چاپ و راننده این ماشین را به اندازه ۱۰۰ جریمه کنید و در غیر این صورت پیام NORMAL RECORDED را چاپ کنید.

مثالی از این درخواست:

```
NEW_RECORD 7741181522 1402/10/15
```

▼ **درخواست BUY_LICENSE**

```
BUY_LICENSE <USERNAME> <CAR_PLATE> <LENGTH> <TIMESTAMP>
```

در این درخواست یک نفر با نام `<USERNAME>` برای ماشین خود با شماره پلاک `<CAR_PLATE>` یک طرح ترافیک `<LENGTH>` روزه در روز `<TIMESTAMP>` می‌خرد.

- `<USERNAME>` یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- `<CAR_PLATE>` یک رشته به طول دقیقاً ۱۰ از ارقام است. این رشته می‌تواند با صفر آغاز شود.
- `<LENGTH>` یک عدد صحیح از ۱ تا ۱۰۰۰ است.
- `<TIMESTAMP>` یک رشته به فرمت `yyyy/mm/dd` است که زمان این اتفاق را نشان می‌دهد.

در صورت رخ دادن هر کدام از حالت‌های زیر دقیقا یکی از پیام‌های زیر را به الویت از بالا به پایین اجرا کنید:

- در صورتی که شخصی با نام `<USERNAME>` در سیستم وجود ندارد پیام `INVALID USERNAME` را چاپ کنید.
- در صورتی که این شخص ماشینی با شماره پلاک `<CAR_PLATE>` ندارد پیام `INVALID CAR PLATE` را چاپ کنید.
- قیمت خرید هر طرح برای **یک روز** ۷۰ است و این مبلغ از حساب راننده کسر می‌شود. در صورتی که راننده چنین مبلغی در حساب خود ندارد پیام `NO ENOUGH MONEY` را چاپ کنید.
- در غیر این صورت پیام `BUY LICENSE DONE` را چاپ کنید.

توجه کنید خرید طرح ترافیک برای یک خودرو در روز dام به مدت k روز یعنی این راننده می‌تواند در روزهای $d+1, d+2, d+3, \ldots, d+k$ در محدوده طرح ترافیک تردد کند.

همچنین توجه کنید اگر یک راننده قبلاً طرحی برای ماشین خود برای این روزها خریده باشد به روزهای این طرح اضافه نمی‌شود بلکه برای یک روز دو طرح دارد.

مثالی از این درخواست:

```
BUY_LICENSE alireza 7741181522 13 1402/10/15
```

▼ درخواست `ADD_BALANCE`

```
ADD_BALANCE <USERNAME> <AMOUNT> <TIMESTAMP>
```

در این درخواست فردی با نام <USERNAME> اعتبار حساب خود در این سامانه را به اندازه <AMOUNT> در لحظه <TIMESTAMP> افزایش می‌دهد.

- <USERNAME> یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- <AMOUNT> یک عدد طبیعی کوچک‌تر یا مساوی ۱۰۰۰ است.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

اگر شخصی با نام <USERNAME> در این سامانه وجود ندارد پیام INVALID USERNAME و در غیر این صورت پیام ADD BALANCE DONE را چاپ کنید.

مثالی از این درخواست در ورودی:

```
ADD_BALANCE alireza 850 1402/10/15
```

▼ درخواست GET_BALANCE

```
GET_BALANCE <USERNAME> <TIMESTAMP>
```

در این درخواست فردی با نام <USERNAME> اعتبار حساب خود در این سامانه را در لحظه <TIMESTAMP> می‌پرسد.

- <USERNAME> یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

مثالی از این درخواست:

```
GET_BALANCE alireza 1402/10/15
```

اگر شخصی با نام <USERNAME> در این سامانه وجود ندارد پیام INVALID USERNAME و در غیر این صورت مقدار اعتبار حساب این شخص را چاپ کنید.

▼ **درخواست GET_PENALTY**

```
GET_PENALTY <USERNAME> <TIMESTAMP>
```

در این درخواست مقدار جریمه‌های <USERNAME> در لحظه <TIMESTAMP> را چاپ کنید.

- <USERNAME> یک رشته به طول حداقل ۱ و حداکثر ۲۰ از حروف کوچک و بزرگ انگلیسی و ارقام است.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

اگر شخصی با نام <USERNAME> در این سامانه وجود ندارد پیام INVALID USERNAME و در غیر این صورت مقدار جریمه‌ی این شخص را چاپ کنید.

مثالی از این درخواست:

```
GET_PENALTY alireza 1402/10/15
```

▼ **درخواست GET_LICENSE_DEADLINE**

```
GET_LICENSE_DEADLINE <CAR_PLATE> <TIMESTAMP>
```

در این درخواست اولین روز بعد از <TIMESTAMP> که ماشین با پلاک <CAR_PLATE> طرح ندارد را چاپ کنید. (این موضوع ربطی به زوج و فرد بودن پلاک ندارد.)

- <CAR_PLATE> یک رشته به طول دقیقاً ۱۰ از ارقام است. این رشته می‌تواند با صفر آغاز شود.
- <TIMESTAMP> یک رشته به فرمت yyyy/mm/dd است که زمان این اتفاق را نشان می‌دهد.

اگر ماشینی با این شماره پلاک در سیستم ثبت نشده است پیام INVALID CAR PLATE را چاپ کنید در غیر این صورت اولین روزی که بعد از روز <TIMESTAMP> این ماشین طرح ندارد را به صورت yyyy/mm/dd چاپ

کنید.

توجه کنید اگر این ماشین تاکنون طرح نداشته است باید فردای `<TIMESTAMP>` را چاپ کنید.

مثالی از این درخواست:

```
GET_LICENSE_DEADLINE 7741181522 1402/10/15
```

نکات

- این سامانه برای سال ۱۴۰۰ به بعد طراحی شده است یعنی تمام زمان‌های داده شده در سال ۱۴۰۰ یا بعد از آن است. همچنین می‌دانیم ۱۴۰۰/۰۱/۰۱ شنبه است.
- منظور از روزهای زوج، شنبه، دوشنبه و چهارشنبه و منظور از روزهای فرد، یک‌شنبه، سه‌شنبه، پنج‌شنبه است. روز جمعه نه زوج است و نه فرد و تردد ماشین‌ها برای این روز آزاد است.
- در این سوال ماه‌ها ۳۰ روزه هستند و سال کبیسه نداریم.
- توجه کنید پول داخل حساب هیچ ارتباطی با مبالغ جریمه شده ندارد.

ورودی

در هر سطر از ورودی یکی از درخواست‌ها که در متن سوال آمده است را دریافت می‌کنید. تعداد درخواست‌ها حداکثر ۳۰۰ است.

بعد از آخرین درخواست برای نشان دادن پایان درخواست‌ها کلمه `END` چاپ می‌شود.

تضمین می‌شود تمام `TIMESTAMP` ها قابل قبول و متمایز و به ترتیب زمان صعودی باشد.

خروجی

در هر سطر از خروجی پاسخ دستوری را که در ورودی دریافت کرده‌اید چاپ می‌کنید.

مثال‌ها

ورودی نمونه ۱

```
REGISTER amin 1400/01/01
REGISTER ali 1400/01/07
REGISTER amin 1400/01/18
REGISTER_CAR amin 7124862189 1400/01/21
REGISTER_CAR amin 2564518722 1400/01/23
REGISTER_CAR amin 5654121543 1400/01/24
NEW_RECORD 7124862189 1400/01/27
NEW_RECORD 7124862189 1400/01/28
NEW_RECORD 7124862189 1400/01/29
GET_PENALTY amin 1400/02/04
GET_BALANCE amin 1400/02/05
ADD_BALANCE amin 150 1400/02/06
GET_PENALTY amin 1400/02/07
GET_BALANCE amin 1400/02/08
GET_LICENSE_DEADLINE 2564518722 1400/02/09
END
```

خروجی نمونه ۱

```
REGISTER DONE
REGISTER DONE
INVALID USERNAME
REGISTER CAR DONE
REGISTER CAR DONE
REGISTER CAR DONE
NORMAL RECORDED
NORMAL RECORDED
PENALTY RECORDED
100
0
ADD BALANCE DONE
100
150
1400/02/10
```

ورودی نمونه ۲

```
REGISTER amin 1400/01/02
REGISTER ali 1400/01/07
REGISTER amin 1400/01/08
REGISTER_CAR amin 7124862189 1400/01/09
REGISTER_CAR amin 2564518722 1400/01/10
REGISTER_CAR amin 5654121543 1400/01/14
ADD_BALANCE amin 350 1400/01/19
GET_BALANCE amin 1400/01/20
BUY_LICENSE amin 2564518722 3 1400/01/22
GET_LICENSE_DEADLINE 2564518722 1400/02/03
GET_BALANCE amin 1400/01/24
NEW_RECORD 2564518722 1400/01/27
NEW_RECORD 2564518722 1400/01/28
NEW_RECORD 2564518722 1400/01/29
GET_PENALTY amin 1400/02/04
GET_BALANCE amin 1400/02/08
END
```

خروجی نمونه ۲

```
REGISTER DONE
REGISTER DONE
INVALID USERNAME
REGISTER CAR DONE
REGISTER CAR DONE
REGISTER CAR DONE
ADD BALANCE DONE
350
BUY LICENSE DONE
1400/02/04
140
PENALTY RECORDED
NORMAL RECORDED
NORMAL RECORDED
```

```
100
140
```

در صورت نیاز به راهنمایی به تمپلیت های زیر مراجعه کنید . توجه کنید استفاده از توابع در این سوال بسیار اهمیت دارد ! در صورت رعایت نکردن فانکشنالیتی پاسخ شما رد خواهد شد !

▼ الگوریتم

1. آماده‌سازی داده‌ها (Initialization)

- تعریف ساختارهای داده‌ای برای ذخیره اطلاعات:
  - **کاربران:** شامل موجودی حساب و مجموع جریمه‌ها.
  - **خودروها:** شامل مالک خودرو و طرح‌های خریداری‌شده.
  - **طرح‌های ترافیک:** تاریخ شروع و مدت زمان اعتبار هر طرح برای هر خودرو.

2. خواندن ورودی‌ها (Input Handling)

- خواندن دستورات ورودی خط به خط.
- تا زمانی که دستور `END` دریافت نشده، ادامه بده.
- هر خط را به اجزای دستور (کلمه کلیدی و پارامترها) تقسیم کن.

3. پردازش دستورات (Command Processing)

- بررسی کن که دستور از چه نوعی است و تابع مربوطه را فراخوانی کن:

۱. **REGISTER <USERNAME> <TIMESTAMP>**

- ثبت‌نام کاربر جدید.
- بررسی یکتایی نام کاربری.
- ایجاد حساب جدید با موجودی و جریمه صفر.

۲. **REGISTER_CAR <USERNAME> <CAR_PLATE> <TIMESTAMP>**

- ثبت خودرو برای کاربر.
- بررسی وجود کاربر و یکتا بودن پلاک خودرو.

- اضافه کردن خودرو به لیست خودروهای کاربر.

۳. **ADD_BALANCE <USERNAME> <AMOUNT> <TIMESTAMP>**

- افزایش موجودی حساب کاربر.
- بررسی وجود کاربر.
- اضافه کردن مبلغ به موجودی حساب.

۴. **BUY_LICENSE <USERNAME> <CAR_PLATE> <LENGTH> <TIMESTAMP>**

- خرید طرح ترافیک برای خودرو.
- بررسی مالکیت خودرو، موجودی حساب و کسر مبلغ.
- ثبت تاریخ‌های اعتبار طرح برای خودرو.

۵. **NEW_RECORD <CAR_PLATE> <TIMESTAMP>**

- ثبت ورود خودرو به محدوده طرح.
- بررسی تطابق روز (زوج یا فرد بودن) با پلاک.
- بررسی خرید طرح برای روز موردنظر.
- ثبت جریمه در صورت تخلف.

۶. **GET_BALANCE <USERNAME> <TIMESTAMP>**

- نمایش موجودی حساب کاربر.
- بررسی وجود کاربر و چاپ موجودی.

۷. **GET_PENALTY <USERNAME> <TIMESTAMP>**

- نمایش مجموع جریمه‌های ثبت‌شده برای کاربر.
- بررسی وجود کاربر و چاپ میزان جریمه.

۸. **GET_LICENSE_DEADLINE <CAR_PLATE> <TIMESTAMP>**

- پیدا کردن اولین روز بدون طرح ترافیک برای خودرو بعد از تاریخ مشخص.
- بررسی ثبت بودن خودرو و محاسبه تاریخ پایان آخرین طرح.

4. مدیریت زمان و تاریخ (Date Handling)

- تبدیل تاریخ‌های ورودی به فرمتی قابل پردازش.
- محاسبه روز هفته برای تشخیص روزهای زوج و فرد:

- شنبه، دوشنبه، چهارشنبه = زوج
- یکشنبه، سه‌شنبه، پنجشنبه = فرد
- جمعه آزاد است.

- مدیریت افزایش تاریخ برای بررسی طرح‌های خریداری‌شده.

5. تولید خروجی (Output Generation)

- بعد از پردازش هر دستور، پیام مناسب را چاپ کن:
  - پیام‌های موفقیت مثل `ADD BALANCE DONE` ، `REGISTER DONE`
  - پیام‌های خطا مثل `NO ENOUGH MONEY` ، `INVALID USERNAME`
  - اطلاعات مالی و جریمه مثل مبلغ موجودی یا میزان جریمه‌ها.

6. پایان برنامه (Termination)

- وقتی دستور `END` دریافت شد، اجرای برنامه را متوقف کن.

▼ توابع

برای پیاده‌سازی سیستم مدیریت طرح زوج و فرد، ابتدا توابع اصلی را که هرکدام مسئول پردازش یک نوع درخواست هستند، تعریف می‌کنیم. هر تابع ورودی‌های مشخصی دارد و باید اعتبارسنجی‌های لازم را انجام دهد.

1. تابع `register`

کاربرد:

- ثبت‌نام کاربر جدید در سیستم.
- اگر کاربر قبلاً ثبت شده باشد، پیام خطا می‌دهد.
- در صورت موفقیت، موجودی اولیه و جریمه را روی صفر تنظیم می‌کند.

```
def register(username):
# بررسی کن که آیا نام کاربری قبلاً ثبت شده یا نه
```

```
    # اگر ثبت شده بود پیام "INVALID USERNAME" چاپ کن
    # اگر ثبت نشده بود، کاربر جدید را اضافه کن و موجودی و جریمه را صفر کن
    # پیام موفقیت‌آمیز "REGISTER DONE" را چاپ کن
        pass
```

2. تابع register_car

کاربرد:

- ثبت یک خودرو برای کاربر.
- بررسی می‌کند که کاربر وجود دارد یا خیر.
- اگر پلاک قبلاً ثبت شده باشد، پیام خطا می‌دهد.
- در غیر این صورت، خودرو را به کاربر نسبت می‌دهد.

```
def register_car(username, plate):
    # بررسی کن که آیا کاربر وجود دارد یا نه
    # بررسی کن که پلاک قبلاً ثبت نشده باشد
    # اگر همه چیز درست بود، خودرو را ثبت کن و مالک آن را مشخص کن
    # پیام "REGISTER CAR DONE" را چاپ کن
        pass
```

3. تابع new_record

کاربرد:

- ثبت ورود یک خودرو به محدوده طرح ترافیک.
- بررسی می‌کند که خودرو اجازه ورود داشته یا خیر.
- در صورت تخلف، جریمه ثبت می‌کند.

```
def new_record(plate, date):
    # بررسی کن که آیا پلاک ثبت شده یا نه
    # اگر ثبت نشده بود پیام "INVALID CAR PLATE" چاپ کن
    # بررسی کن که آیا خودرو مجوز طرح دارد یا خیر
# اگر مجوز ندارد و تخلف صورت گرفته بود، جریمه ثبت کن و پیام "PENALTY RECORDED" چاپ کز
```

```
    # اگر تخلفی نبود پیام "NORMAL RECORDED" چاپ کن
    pass
```

4. تابع buy_license

کاربرد:

- خرید طرح ترافیک برای خودرو توسط کاربر
- بررسی می‌کند که کاربر و خودرو وجود دارند و مالکیت درست است.
- بررسی می‌کند که کاربر اعتبار کافی دارد یا نه.
- در صورت موفقیت، طرح را ثبت می‌کند.

```
def buy_license(username, plate, length, date):
    # بررسی کن که آیا کاربر ثبت شده یا نه
    # بررسی کن که پلاک ثبت شده و متعلق به کاربر هست یا نه
    # محاسبه هزینه طرح بر اساس تعداد روز
    # بررسی کن که آیا کاربر اعتبار کافی دارد یا نه
    # در صورت تأیید، مبلغ را کسر کن و مجوز طرح را ثبت کن
    # پیام "BUY LICENSE DONE" را چاپ کن
    pass
```

5. تابع add_balance

کاربرد:

- افزایش موجودی حساب کاربر.
- بررسی می‌کند که کاربر وجود دارد یا خیر.
- در صورت موفقیت، مبلغ را به موجودی کاربر اضافه می‌کند.

```
def add_balance(username, amount):
    # بررسی کن که آیا کاربر ثبت شده یا نه
    # اگر کاربر وجود داشت، مبلغ را به حساب او اضافه کن
    # پیام "ADD BALANCE DONE" را چاپ کن
    pass
```

6. تابع `get_balance`

کاربرد:

- نمایش موجودی حساب کاربر.
- بررسی می‌کند که کاربر وجود دارد یا خیر.
- در صورت وجود، موجودی را نمایش می‌دهد.

```
def get_balance(username):
# بررسی کن که آیا کاربر ثبت شده یا نه
# اگر کاربر وجود داشت، موجودی حساب را چاپ کن
    pass
```

7. تابع `get_penalty`

کاربرد:

- نمایش میزان جریمه‌های ثبت‌شده برای کاربر.
- بررسی می‌کند که کاربر وجود دارد یا خیر.
- در صورت وجود، مجموع جریمه‌ها را نمایش می‌دهد.

```
def get_penalty(username):
# بررسی کن که آیا کاربر ثبت شده یا نه
# اگر کاربر وجود داشت، مقدار جریمه‌های ثبت‌شده را چاپ کن
    pass
```

8. تابع `get_license_deadline`

کاربرد:

- پیدا کردن اولین روزی که خودرو بعد از یک تاریخ مشخص طرح ترافیک ندارد.
- بررسی می‌کند که پلاک ثبت شده است یا نه.
- تاریخ پایان آخرین طرح را نمایش می‌دهد.

```
def get_license_deadline(plate, date):
# بررسی کن که آیا پلاک ثبت شده یا نه
# اگر ثبت شده بود، بررسی کن که تا چه تاریخی خودرو طرح دارد
# اولین تاریخی که خودرو طرح ندارد را پیدا کن و نمایش بده
    pass
```