Space Efficient SSet

- محدودیت زمان: 4 ثانیه
- محدودیت حافظه: 200مگابایت

ساختمان داده **Skiplist** را در کلاس بحث کردیم و نشان دادیم چگونه می‌توان SSet را با آن پیاده سازی کرد. بحث کردیم که از لحاظ میزان مصرف حافظه SSet-Skiplist میزان $O(n)$ فضا مصرف می‌کند. همچنین در کلاس ساختمان داده **SEList** را پیاده سازی کردیم. از ایده SEList استفاده کنید و ساختمان داده SSet-Skiplist را به گونه‌ای بهینه پیاده‌سازی کنید که میزان حافظه هدر رفته آن در هر لحظه $O(\sqrt{n})$ باشد.

در این تمرین هر عنصر مجموعه SSet دو مقدار زیر را دارد و بر اساس key مرتب شده است:

- یک کلید (key) که یک string است.
- یک مقدار (value) که یک عدد double است.

این ساختمان داده جدید تمام عملیات‌های SSet را در زمان $O(\log n)^E + O(\sqrt{n})$ پشتیبانی می‌کند:

- عملیات find (key) که مقدار value عنصری را که کلید آن مساوی یا کوچکترین کلید بزرگتر از key است را برمی گرداند. پس اگر چنین عددی وجود داشت، خروجی یک double است، در غیر این صورت رشته "not found" را چاپ کند.
- عملیات add (key, value) که عنصری با کلید key و مقدار value را، اگر کلید تکراری نباشد، به مجموعه اضافه می‌کند. اگر کلید تکراری نبود این عملیات رشته "added" را چاپ کند و اگر قبلا چنین کلیدی در مجموعه موجود بود رشته "already in there" را چاپ کند.
- عملیات remove (key) که عنصری با کلید key را حذف می‌کند اگر وجود داشته باشد. اگر کلید وجود داشت این عملیات رشته "removed" را چاپ کند و اگر چنین کلیدی در مجموعه موجود نبود رشته "does not exist" را چاپ کند.

توجه کنید هر چند میتوانید از جاوا و پایتون نیز استفاده کنید اما توصیه نمی‌شود چون کارایی بالایی برای این تمرین مدنظر است. ریسک پیاده‌سازی با این دو زبان به عهده دانشجو است.

ورودی و خروجی

در ابتدا مجموعه SESSet خالی است. در خط اول تعداد عملیات‌ها داده شده است.

سپس در خط های بعدی تعدادی عملیاتfind و add و remove قرار داده شده است. ابتدای هر خط با کلمه add یا remove یا find شروع می شود:

- اگر خطی با کلمه add شروع شده باشد بعد از یک فاصله یک رشته که فاصله در آن وجود ندارد به عنوان key داده شده است. بعد از یک فاصله یک عدد double قرار دارد که مقدار value را مشخص می‌کند. اگر کلید تکراری نبود این عملیات عنصر را در جای مناسب خودش در مجموعه اضافه کرده و رشته "added" را چاپ کند و اگر قبلا چنین کلیدی در مجموعه موجود بود رشته "already in there" را چاپ کند.
- اگر خطی با کلمه remove شروع شده باشد، بعد از یک فاصله یک رشته که فاصله در آن وجود ندارد به عنوان key داده شده است. اگر کلید وجود داشت این عملیات عنصر مربوطه را حذف می‌کند و رشته "removed" را چاپ کند و اگر چنین کلیدی در مجموعه موجود نبود رشته "does not exist" را چاپ کند.
- اگر خطی با کلمه find شروع شده باشد، بعد از یک فاصله یک رشته که فاصله در آن وجود ندارد به عنوان key داده شده است. مقدار value عنصری را که کلید آن مساوی یا کوچکترین کلید بزرگتر از key است را برمی گرداند که یک double است، در غیر این صورت رشته "not found" را چاپ کند.

تعداد عملیات‌های ساختمان داده در هیچ کدام از تست ها بیش از 10^4 نخواهد بود.

توجه داشته باشید هر خروجی خواسته شده برای هر عملیات در یک خط مستقل چاپ شود.

مثال

ورودی نمونه ۱

```
6
add x02msd2 2.0
add x02msd2 2.1
add j9sjd0 2.0
```

```
find x02msd
remove xsx
remove j9sjd0
```

خروجی نمونه ۱

```
added
already in there
added
2.0
does not exist
removed
```

ورودی نمونه ۲

```
8
remove x0
add x0 0.0
add x1 1.2
add x5 1.1
find x2
remove x2
remove x5
find x2
```

خروجی نمونه ۲

```
does not exist
added
added
added
1.1
does not exist
removed
not found
```