درخت AVL

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: 100 مگابایت

هدف این تمرین یادگیری درخت AVL است. این تمرین نسبت به مطالبی که در کلاس مطالعه کردیم بسیار ساده تر می باشد. در کلاس ما دو درخت جستجوی دودویی که ارتفاع لگاریتمی داشتند را مطالعه کردیم (تریپ و درخت قرمز و سیاه). در این تمرین یک نوع دیگر از درخت جستجوی دودویی با نام AVL پیاده سازی خواهیم کرد.

درخت AVL ارتفاع لگاریتمی و همچنین بالانس بودن درخت را گارانتی میکند. از منابع زیر این درخت را مطالعه کنید و عملیات های add ،find و remove را برای این درخت پیاده سازی نمایید.

https://en.wikipedia.org/wiki/AVL_tree

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

http://www.geeksforgeeks.org/avl-tree-set-1-insertion/

http://www.geeksforgeeks.org/avl-tree-set-2-deletion/

همه منابع، کد را برای پیاده سازی دارند. می توانید آنها را مطالعه کنید تا برای پیاده سازی ایده بگیرید. **دقت کنید که از جایی کد کپی نکنید** چون ما میزان شباهت کد شما را در زبان های مختلف با کدهایی از منابع مختلف می سنجیم. تغییرات کوچک در برنامه مانند عوض کردن نام متغییر یا جابجا کردن توابع در میزان شباهت تغییری ایجاد نمیکند. جریمه کپی کردن کد برابر است با صفر شدن در تمام درس.

ورودی و خروجی

در ابتدا درخت خالی است. در خط اول تعداد عملیات‌ها داده شده است.

سپس در خط های بعدی تعدادی عملیات find 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 عدد double قرار دارد که مقدار value را مشخص می‌کند. اگر کلید تکراری نبود این عملیات عنصر را در جای مناسب خودش در درخت 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 عضو جدید به درخت اضافه شد و نیازی به بالانس کردن درخت AVL بود آنگاه در خط بعد "balancing" را چاپ می کند و در همان خط گره ای را که اختلاف زیر درختانش بیش از 1 است و نیاز به بالانس دارد را چاپ کند.

- اگر خطی با کلمه remove شروع شده باشد، بعد از یک فاصله یک عدد double قرار دارد که اگر در درخت وجود داشت این عملیات عنصر مربوطه را حذف می‌کند و رشته "removed" را چاپ کند و اگر چنین عنصری در مجموعه موجود نبود رشته "does not exist" را چاپ کند. اگر نیازی به بالانس کردن درخت AVL بود آنگاه در خط بعد "balancing" را چاپ می کند و در همان خط گره ای را که اختلاف زیر درختانش بیش از 1 است و نیاز به بالانس دارد را چاپ کند.
- اگر خطی با کلمه find شروع شده باشد، بعد از یک فاصله یک عدد double قرار دارد. مقدار value عنصری را که مساوی یا کوچکترین عنصر بزرگتر از آن است را برمی گرداند، 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 2.0
add 2.1
add 2.2
find 3.0
remove 1.0
remove 2.1
```

خروجی نمونه ۱

```
added
added
added
balancing 2.0
not found
does not exist
removed
```

ورودی نمونه ۲

```
8
remove 1.0
add 1.0
```

```
add 2.0
add 3.0
find 2.0
remove 1.0
remove 1.0
find 1.0
```

خروجی نمونه ۲

```
does not exist
added
added
added
balancing 1.0
2.0
removed
does not exist
2.0
```