Crimes of Ranrok

- محدودیت زمان: نامحدود
- محدودیت حافظه: 256 مگابایت

در دنیایی که جادوگران میان بی‌جادویان (ماگل‌ها) به طور مخفی زندگی می‌کنند، بانک گرینگاتز که از اجسام و ارز جادویی محافظت می‌کند، بخشی از بانک را به ماگل‌هایی که از جادو باخبرند و با جادوگران زندگی می‌کنند اختصاص داده است. خبر رسیده است که گابلین یاغی‌ای به نام رنراک که قصد پایان دادن به سلطه‌ی جادوگران بر گابلین‌ها را دارد، به گرینگاتز حمله کرده است. بر اثر این حمله، مسئول بخش ماگل‌های گرینگاتز کشته شده و فایل‌های ثبت داده‌های این بخش خراب شده‌اند.

رئیس گرینگاتز که درگیر مدیریت این بحران عظیم و رفع نفوذ پذیری بانک است، از شما خواسته است برای کمک به او نرم‌افزاری طراحی کنید که داده‌ها را تصحیح کرده، پایگاه داده‌ی بانک را از نو بسازد و برای یک روز مدیریت خودکار بخش ماگل‌های بانک را به دست بگیرد. برای اینکه بتوانید بخش‌های مورد نیاز را در نرم‌افزار خود تعبیه کنید، رئیس بانک در پیامی کارکرد بخش ماگل‌های گرینگاتز را به شکل زیر شرح داده است.

ساختار بانک گرینگاتز

در گرینگاتز، هر حساب معادل یک سردابه در اعماق غارهاست. هر سردابه توسط یک کلید جادویی باز می‌شود که بر آن یک شماره‌ی حساب پنج رقمی درج شده است و از آن دو نسخه موجود است؛ یک نسخه در دست صاحب حساب و دیگری در دست مسئول بانک می‌باشد. زمانی که کارخواهی وارد بانک می‌شود، با ارائه‌ی کلید (شماره حساب) به کارکنان، توسط یک گابلین به سردابه‌ی حساب خود منتقل می‌شود.

هر کارخواه می‌تواند پس از ورود به سردابه‌ی خود، اعمال زیر را انجام دهد:

۱. موجودی خود را مشاهده کند.
۲. به موجودی خود اضافه کند.
۳. از موجودی خود برداشت کند.
۴. از گابلین همراه خود بخواهد مقداری از موجودیش را به سردابه‌ی دیگری منتقل کند.
۵. درخواست نابودی سردابه و حذف حساب خود را بدهد.
۶. از سردابه خارج شود و کلیدش را پس بگیرد.

همچنین به علت تمرکز گرینگاتز بر سرویس‌دهی به جادوگران، تنها تعداد 9 مشتری ماگل در روز پذیرفته می‌شوند و پس از این تعداد، بانک برای ماگل‌ها تعطیل می‌شود. بر این اساس لازم است شما برنامه‌ای بنویسید که علاوه بر بازسازی پایگاه داده‌ی بانک از داده‌های خراب شده‌ی قبل، انجام تمامی این مراحل را برای 9 بار ورود و خروج به حساب با تنها کار با نرم‌افزار ممکن سازد. یعنی برنامه‌ی شما بایستی به نوع دلخواهتان بخش‌های زیر را امکان پذیر کند.

بخش‌های برنامه

نکته: در شیوه‌ی نوشتن و حتی تغییر ترکیب بخش‌های زیر آزادی کامل دارید؛ مدل و مثال‌های ذکر شده در ادامه تنها برای کمک به درک نیازهای برنامه است. در مدل‌سازی دلخواه برنامه‌ی خود تنها بایستی به موارد زیر دقت کنید:

۱. حتما دو ورودی داده‌ها که به شکل ذکر شده خراب شده‌اند را در ابتدا دریافت کنید.
۲. نوع داده‌های شماره حساب و موجودی را حفظ کنید.
۳. ساختار ذکر شده‌ی بخش ماگل‌های گرینگاتز را رعایت کنید.
۴. امکان تعامل با منو، حرکت در آن و ارسال درخواست‌های متعدد را تعبیه کنید.
۵. برای هر قابلیت برنامه، تمام حالات ممکن را برای ورودی‌ها در نظر گرفته و در صورت نامناسب بودن ورودی عملی برای برنامه طراحی کنید.

بخش اولیه: تصحیح داده‌ها

برنامه‌ی شما باید در ابتدا پایگاه داده‌ی بانک را از نو بسازد. برای این کار بایستی دو دسته داده‌ی نیاز به پاک‌سازی شماره‌ی حساب‌ها و موجودی‌ها را (که هر دو رشته‌ای نیاز به پردازش هستند) در ورودی دریافت کند. این داده‌ها هر کدام به شکلی خراب شده‌اند.

۱. **شماره حساب‌ها:** تمامی شماره حساب‌ها (که هر کدام یک عدد صحیح پنج رقمی هستند) به صورت اعداد کنار هم چسبانده شده‌اند و حروف انگلیسی کوچک میانشان پخش شده است. برای مثال، اگر شماره حساب‌های بانک شماره‌های 16439، 92384، 73021، 80475 و 20054 باشند، ورودی خراب شده‌ی شماره حساب‌ها به شکل زیر است.

```
q1yu6a4s3fd9nj9efjkh2nl3mj8vg47m3ed0i2ow1d80x47e5200f5g4
```

۲. **موجودی‌ها:** تمامی موجودی‌ها (که هر کدام یک عدد اعشاری از 0 تا 10^{10} هستند) کنار هم قرار داده شده‌اند، به طوری که هر موجودی از موجودی بعد خود توسط یک خط فاصله ($-$) جدا شده است. همچنین هر رقم موجودی به معادل نوشتاری انگلیسی آن با حروف کوچک تبدیل شده است (یعنی رقم 1 به شکل one، رقم 2 به شکل two و غیره نوشته شده‌اند و به جای نقطه‌ی ممیز نیز dot قرار داده شده است) و این ارقام نوشتاری با ویرگول از هم جدا شده‌اند. برای مثال، اگر موجودی‌های معادل حساب‌های ذکر شده در بالا به ترتیب برابر 25.2، 1398، 0.05، 6745 و 0 گالیون باشند، ورودی خراب شده‌ی دوم برنامه برای موجودی‌ها به شکل زیر است.

```
two,five,dot,two-one,three,nine,eight-zero,dot,zero,five-six,seven,fc
```

در نهایت این دو رشته باید توسط برنامه‌ی شما تمیز شده و به اعداد معادلشان تبدیل شوند تا دیکشنری پایگاه داده ساخته شود. در دیکشنری بایستی هر شماره حساب و موجودی مربوطه به عنوان (به ترتیب) کلید و مقدار قرار گیرند. برای مثال، دیکشنری حاصل از تصحیح داده‌های مثال بالا به صورت زیر می‌باشد (نیازی نیست دیکشنری تولیدی در خروجی چاپ شود).

```
{16439: 25.2, 92384: 1398, 73021: 0.05, 80475: 6745, 20054: 0}
```

بخش ثانویه: ورود به حساب

پس از تشکیل پایگاه داده، برنامه‌ی شما بایستی آماده‌ی سرویس‌دهی به ماگل‌ها باشد و به یک نفر در صف اجازه‌ی ورود دهد؛ یعنی مثلا به یک کارخواه جدید خوش‌آمد گفته و با یک جمله از او کلید (عدد صحیح پنج رقمی شماره حساب) را به عنوان ورودی درخواست کند.

نکته: از آنجا که شماره حساب همان کلید جادویی ارائه شده است که غیر قابل جعل یا خرابیست، فرض کنید که هیچ کارخواهی شماره حساب فرد دیگری را به برنامه وارد نمی‌کند؛ یعنی امنیت بانک با جادو تضمین شده است و دریافت شماره حساب تنها برای تعیین سردابه (حساب) اوست. همچنین دقت کنید این عمل بایستی برای کارخواه‌های جدید به ترتیب تکرار شود و هرگاه 9 بار سرویس‌دهی انجام شده بود، بانک با چاپ جمله‌ای تعطیل شود.

برای مثال، فرض کنید دو ورودی ذکر شده در بخش اولیه دریافت شده‌اند. آنگاه برنامه پس از تشکیل پایگاه داده خطوط زیر را نمایش می‌دهد و منتظر دریافت کلید (شماره حساب) می‌ماند.

```
Welcome to the Gringotts Wizarding Bank!
Please present your vault key:
```

آنگاه در صورت دریافت کلید (برای نمونه 80475) به منوی کاربر بانک (بخش ثالثیه) می‌رود. همچنین پس از خروج کاربر از منوی بخش ثالثیه (کار با حساب خود) در صورت اتمام حجم سرویس‌دهی بانک، برنامه جمله‌ی زیر را چاپ می‌کند و دیگر ورودی‌ای از کاربران جدید دریافت نمی‌کند.

```
Sorry, Gringotts is closed for the day. See you tomorrow!
```

بخش ثالثیه: منوی تعاملی

در صورت عدم اتمام حجم سرویس‌دهی و دریافت یک شماره حساب در بخش ثانویه، برنامه بایستی وارد منوی تعاملی کارخواه شود و قابلیت‌های ذکر شده در صورت ورود او به سردابه‌اش را هر چند بار که خواست در دسترس او قرار دهد.

نکته: برای هر قابلیت، موقعیت واقعی و احتمالات مربوطه‌ی هر کدام (مثلا برداشت بیشتر از موجودی، انتقال وجه به شماره حساب ناموجود، وارد کردن عدد منفی در زمان برداشت از حساب و در نتیجه اضافه کردن به موجودی به علت وجود دو منفی و غیره) را تا حد امکان در نظر بگیرید.

برای مثال، برنامه می‌تواند منویی برای کاربر چاپ کند که در آن به هر عمل عدد صحیحی اطلاق شده است. سپس عدد عمل مورد نظر کاربر را در ورودی دریافت کند و آن عمل را اجرا کند. اگر فرض کنیم در بخش قبل وارد حساب 80475 شده‌ایم، کاربر دارای حساب، منوی اصلی زیر را مشاهده می‌کند.

```
You're in, 80475! What would you like to do?
1) View Your Balance
2) Deposit Money
3) Withdraw Money
4) Transfer Money
5) Close Your Account
6) Exit Your Account
Enter the relevant number:
```

حال اگر کاربر عدد صحیح 7 را وارد کند، برنامه به او می‌گوید این گزینه ناموجود است و دوباره امتحان کند. و یا اگر عدد صحیح 4 را برای انتقال وجه وارد کند، برنامه از او شماره حساب مقصد و مقدار گالیون انتقالی را

می‌خواهد؛ آنگاه اگر شماره حساب ناموجود بود و یا وجه انتقالی از موجودی بیشتر بود، از او می‌خواهد که دوباره امتحان کند و یا با وارد کردن عدد -1 به منوی اصلی بالا برگردد؛ اگر درست بود نیز تغییرات لازم را در پایگاه داده اعمال می‌کند که اگر فرد صاحب حساب مقصد وارد حساب خود شد، افزایش موجودی را مشاهده کند.

لازم به ذکر است که وجود گزینه‌ی خروج از حساب الزامیست، تا کارخواه دیگری نیز بتواند پس از این کاربر و در صورت عدم اتمام سرویس‌دهی (بخش ثانویه) وارد حساب خود شود.

برنامه‌ی خود را به شیوه‌ی دلخواه برای پاک‌سازی داده‌های ذکر شده و ارائه‌ی سرویس‌های خواسته شده‌ی بانک مدل‌سازی کنید و در مکان مناسب آپلود کنید. تصحیح برنامه به صورت دستی انجام می‌شود.