درس برنامه‌سازی پیشرفته

تمرین دوم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۹۹ـ۹۸

اساتید:

مهدی مصطفی‌زاده، ایمان عیسی‌زاده، امیر ملک‌زاده، علی چکاه

مبحث:

برنامه نویسی شی‌گرا

مهلت ارسال:

۱۲ فروردین

ساعت ۲۳:۵۹:۵۹

ویراستار فنی:

امیرمهدی نامجو و صابر ظفرپور

به موارد زیر توجه کنید:

* به ازای هر سوال در سامانه کوئرا بخش جداگانه ای برای آپلود شما وجود دارد. شما باید تمامی فایل‌های کلاس‌هایی که با فرمت java. نوشته‌اید را در قالب فشرده با فرمت zip. در آورده و در سامانه کوئرا آپلود کنید. توجه کنید که حتما فایل حاوی کلاسی که تابع

```
public static void main(string[] args)
```

در آن قرار دارد و اجرای برنامه از آن جا شروع می‌شود، باید بدون قرار گرفتن در هیچ پوشه‌ای درون فایل zip قرار بگیرد و نام آن هم Main.java باشد. امکان پوشه بندی و ایجاد package های مختلف برای سایر فایل‌ها امکان پذیر است. به عنوان نمونه می‌توانید به ساختار قابل قبولی و درستی که در زیر رسم شده است، توجه کنید. توجه کنید که پوشه src که فایل‌ها در آن قرار دارد، نباید در فایل zip. باشد.

answer.zip
- models
  - Model1.java
  - Model2.java
- views
  - View1.java
- controllers
  - Controller1.java
- Main.java
- MyClass.java
- MyClass2.java

* ورودی و خروجی شما باید عیناً شبیه به نمونه‌های ورودی و خروجی باشد؛ لذا عبارت‌هایی همچون "Enter your number" را قبل از گرفتن ورودی نباید چاپ کنید.

* پس از ارسال فایل مربوط به هر سوال، سامانه‌ی کوئرا به‌صورت لحظه‌ای برنامه‌ی شما را داوری کرده و نمره‌ی آن سوال را به شما اعلام می‌کند که در صورت کم بودن نمره‌تان، می‌توانید آن را تصحیح کرده و دوباره ارسال کنید.
* هم‌فکری و هم‌کاری در پاسخ به تمرینات اشکالی ندارد و حتی توصیه نیز می‌شود؛ ولی پاسخ ارسالی شما باید حتما توسط خود شما نوشته شده‌باشد. در صورت هم‌فکری در مورد یک سوال، نام فرد دیگر را به‌صورت کامنت در ابتدای کد هر سوال بنویسید.
* شما می‌توانید تمامی سوالات و ابهامات خود را در سایت کوئرا در بخش مشخص‌شده برای این تمرین بپرسید.
* مهلت ارسال تمرین تا ساعت ۲۳:۵۹:۵۹ روز ۱۲ فروردین ۱۳۹۹ است.
* به‌ازای هر روز تاخیر در ارسال پاسخ هر سوال، ۳۰ درصد از نمره‌ی کسب‌شده‌ی شما در آن سوال کم می‌شود. به عنوان مثال اگر پاسخ یک سوال را با دو روز تاخیر ارسال کنید، فقط ۴۰ درصد از نمره‌ای که برای آن سوال گرفته‌اید برای شما لحاظ خواهد شد.
* از آن جا که هدف این تمرین ، آشنایی با مفاهیم شی گرایی است، رعایت این موارد همانند رعایت کلین‌کد اجباری است.
* همچنین در سوال اول نیز شما ملزم به رعایت uml داده شده در طراحی خود هستید.

۱ ایزی ایزی

در کتبِ تاریخی متعددی آمده است که در کشور برزیل (از توابع آبادانِ کنونی)، سیستمِ بانکیِ پیشرفته‌ای وجود داشته و مردمِ آنجا (مِن جمله پله‌ی فقید و آقامون رونالدینیو) به داشتنِ چنین سیستمی افتخار می‌کردند.

بر اثر حادثه‌ای تاریخی برزیل از آمریکای جنوبی جدا و به زادگاهِ خود، آبادن برمی‌گردد. متاسفانه در حین عبور از تنگه‌ی هرمز، بر اثر پهناوریِ بیش از حدِ برزیل، قسمتی از آن به تنگه گیر کرده، شکسته و در آب غرق می‌شود. نکته‌ی بد اینجاست که سیستمِ بانکی برزیل در آن قسمتِ غرق شده قرار داشت و اکنون برزیلِ باقیمانده سیستمِ بانکی ندارد.

در این سوال ما از شما می‌خواهیم تا سیستمِ بانکیِ برزیل را مجددا پیاده سازی کنید. خوشبحتانه نقشه‌ی این سیستمِ بانکی از بین نرفته و در این سوال، شما باید دقیقاً طبق همین نقشه (**که در اصل به آن UML میگویند**) سیستم بانکی برزیل را مجددا پیاده سازی کنید.

این سیستم بانکی دارای اجزای زیر است:

بانک: در برزیل هر بانک یک نام مخصوص به خود دارد و خدماتِ افتتاح حساب و اعطای وام را ارائه می‌کند.

مشتری: هر مشتری یک اسم مخصوص به خود دارد و هیچ دو مشتری‌ای اسم یکسان

ندارند. نمره‌ی منفی ویژگیِ دیگریست که مشتری‌ها دارند. در مورد نمره‌ی منفی در قسمتِ وام خواهید خواند..

هر مشتری در ابتدا مقداری پول در گاوصندوق خانه اش دارد. او میتواند یک مقدار از این پول را در هر بانکی در قالبِ یک حساب، سپرده گذاری کند. با تمام شدنِ مدتِ تعیین شده برای حساب، مبلغِ سپرده گذاری شده همراه با سودِ آن به صورت اتوماتیک (!) به گاوصندوقِ مشتری برمیگردد.

اگر مشتری زودتر از مدت تعیین شده برای بازپس گرفتنِ سپرده‌اش به بانک مراجعه کند، تنها مبلغِ اصلی (بدون سود) را دریافت میکند و به گاوصندوقش برمیگرداند.

با پس گرفته شدن پول یک حساب یا به پایان رسیدنِ مدت تعیین شده‌ی حساب، آن حساب از بین میرود.

حساب: سه نوع حساب در بانک های برزیل وجود دارد؛ حساب های کوتاه‌مدت، بلندمدت و ویژه. هر حساب شماره، میزان سپرده‌ی اولیه، درصد سود و مدتِ تعیین شده دارد. پس از سپری شدن زمانِ تعیین شده، درصد سود به علاوه‌ی ۱ شده و در میزان سپرده‌ی اولیه ضرب می‌شود، سپس به گاوصندوق صاحب حساب بازمی‌گردد.

درصد سود حساب های کوتاه مدت، بلندمدت و ویژه به ترتیب ۱۰٪، ۳۰٪ و ۵۰٪ است. نوعِ حساب (کوتاه مدت، بلند مدت و ویژه) و مدتِ حساب در هنگامِ ساختن حساب توسط مشتری تعیین میشود.

نکته‌ی دیگر اینکه شماره‌ی هر حسابی که هر مشتری ایجاد می‌کند، یکی بیشتر از تعداد حساب‌هایی میشود که همان مشتری از بدوِ خلقت تا الان ساخته؛ برای مثال اگر یک مشتری پنجمین حسابِ خود را ایجاد کند، شماره‌ی آن حساب ۵ میشود (یعنی ممکن است در بانکی دو حساب با شماره‌ی یکسان وجود داشته باشد ولی مطمئناً آن دو حساب متعلق به دو نفرِ مختلف هستند.).

وام: هر بانک می‌تواند به مشتریان خود وام پرداخت کند. هر وام درصد سود و مقدار دارد. دو نوع وام داریم: وامِ ۶ قسطه و ۱۲ قسطه. بعد از ثبت درخواست، وام به گاوصندوق مشتری وارد میشود. پرداختِ اقساط از انتهای همان واحد زمانی آغاز می‌شود.

مکانیزم پرداخت اقساط به این صورت است که در هر واحد زمانی یک قسط به مقدار زیر از گاوصندوق مشتری کم می‌شود:

$$\text{مقدار هر قسط} = \frac{\text{مقدار اولیه وام} \times (\text{درصد سود} + ۱)}{\text{تعداد کل اقساط}}$$

اگر مشتری در گاوصندوق خود به مقدار قسط پول نداشته باشد، هیچ مقداری از گاوصندوق او کم نشده، تعداد اقساطِ باقیمانده‌اش ثابت می‌ماند و یک نمره‌ی منفی می‌گیرد.
بدیِ نمره ی منفی این است که اگر مشتری‌ای حداقل پنج نمره منفی داشته باشد دیگر وامی به او تعلق نمی‌گیرد.
به این نکته هم توجه داشته باشید که ترتیبِ کم شدنِ اقساط وام‌ها هم براساس ترتیب به وجود آمدنِ وام‌هاست؛ یعنی مثلاً اگر وامِ x زودتر از وامِ y در ورودی آمده باشد، قسطِ وامِ x زودتر از قسط وامِ y از حسابِ مشتریان کم می‌شود.
عملیات‌هایی که در سیستم بانکی برزیل پشتیبانی می‌شوند به شرح زیر هستند:

```
Add a customer with name A and N unit initial money.
```

افزودن یک مشتری با نامِ A و N واحد پولِ اولیه (که A یک اسمِ یک یا چند بخشی است و N یک عدد طبیعیست).

```
Create bank M.
```

افزودن بانک با نامِ M (که M یک اسم یک یا چند بخشی است (مثلا "$golabi$" یا "$kaka\ khan < 3$" دو اسمِ مجاز هستند).

```
Create a (KOOTAH|BOLAN|VIZHE) account for A in M, with duration T and initial deposit of D.
```

افزودن یک حسابِ کوتاه مدت، بلند مدت یا ویژه در بانکِ M با مدت زمانِ T و میزان سپرده‌ی اولیه‌ی D توسط مشتری A.
اگر بانک با نامِ M وجود نداشت، پیغامِ:

```
In dige banke koodoon keshvarie?
```

را چاپ کنید.

اگر مشتری کمتر از D واحد درون گاوصندوق خود پول داشت، پیغامِ

```
Boro baba pool nadari!
```

رو چاپ کنید.
خطاها را به همین ترتیب چک کنید و خروجی دهید

```
Give A's money out of his account number N.
```

برداشتن تمام پول شخص A از حسابِ شماره‌ی .N
اگر مشتری حساب با همچین شماره‌ایَ نداشت، پیغامِ

```
Chizi zadi?!
```

را چاپ کنید.

```
Pay a X unit loan with Y% interest and (6|12) payments from M to A.
```

پرداخت یک وام با مقدار X واحد و سودِ Y% و اقساط ۶ یا ۱۲ واحدی از بانکِ M به شخصِ .A
چنانچه بانک با نامِ M وجود نداشته باشد، پیغامِ

```
Gerefti maro nesfe shabi?
```

را چاپ کنید.
چنانچه A حداقل ۵ نمره‌ی منفی داشته باشد، پیغامِ:

```
To yeki kheyli vazet bade!
```

را چاپ کنید.

```
Pass time by X unit.
```

از ابتدای خلقت X واحد میگذرد.

```
Print A's GAVSANDOOGH money.
```

میزان پول موجود در گاوصندوق مشتری A را چاپ کنید.

```
Print A's NOMRE MANFI count.
```

تعداد نمره های منفی A را چاپ کنید.

آیا شخصِ A در بانکِ M حساب فعال دارد یا نه؟

```
Does A have active account in M?
```

در صورتی که A در بانکِ M حسابِ فعالی داشته باشد عبارت

```
yes
```

و در غیر این صورت عبارت

```
no
```

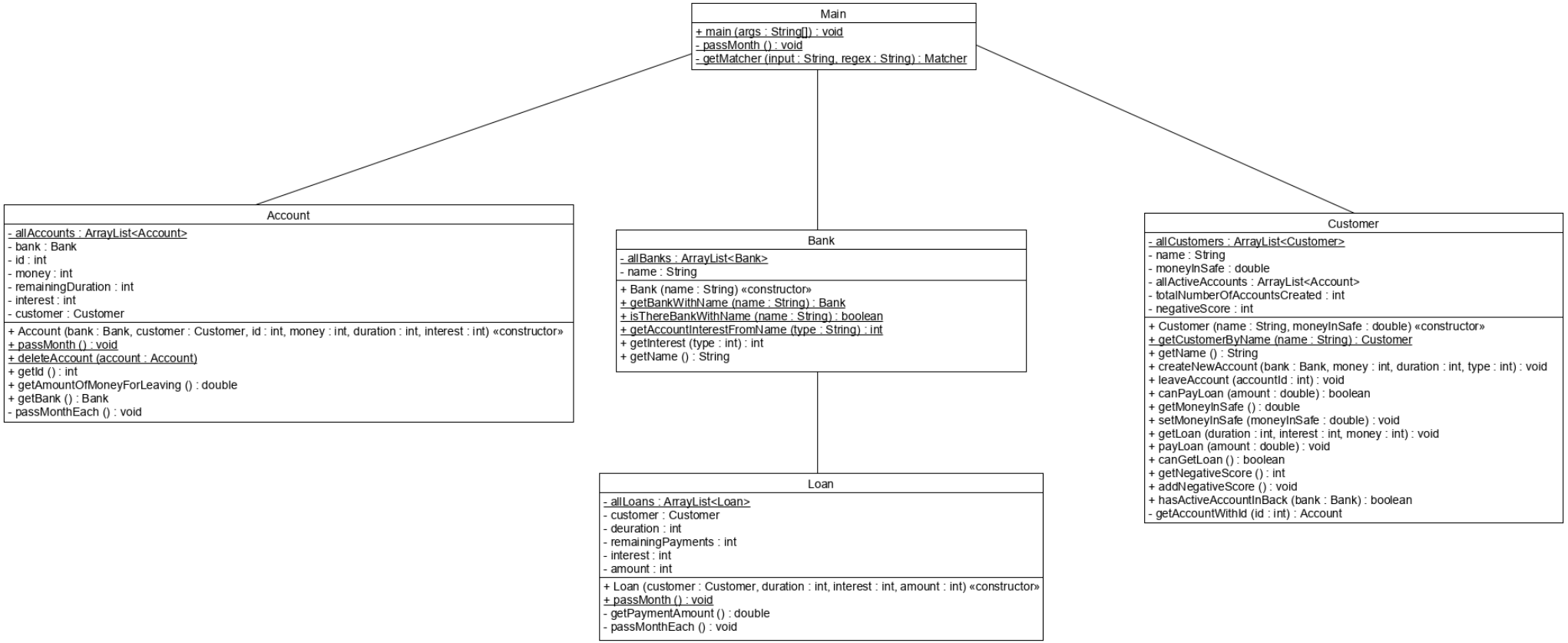
را چاپ کنید.

```
Base dige, berid khonehatoon.
```

انجام عملیات‌های بانکی خاتمه می‌یابد.

- پرداخت اقساط وام و کم شدن از مدتِ تعیین شده برای حساب‌ها، در پایانِ هر واحدِ زمانی اتفاق می‌افتد و سایرِ اعمال از جمله "واریزِ سپرده‌ی حساب‌ها پس از تمام شدنِ مدت حساب"، در ابتدای هر واحدِ زمانی رخ میدهد.
- به نقطه ها و علائم نگارشی آخر دستورات و جملات توجه کنید. در دستورات ورودی و خروجی عیناً همان‌ها آورده می‌شود.

- تضمین می‌شود تنها خطاهایی که در بالا برای آنها پاسخی آمده رخ میدهند و خطاهای دیگری در دستورات ورودی رخ نمیدهد. ترتیب خطاها را همانگونه که آورده شده چک کنید و خروجی دهید.
- در هنگام چاپ اعداد در خروجی، آنها را قطع و چاپ کنید؛ مثلا اگر میزان پول موجود در گاوصندوقِ مشتری A، ۱۲.۳۴ واحد باشد، شما باید عدد ۱۲ را چاپ کنید.
- کدی که میزنید حتماً باید براساس UML داده شده در صفحه بعد پیاده سازی شود. این موضوع در تحویل حضوری بررسی خواهد شد. فایل این تصویر نیز به طور جداگانه قرار گرفته است.


Main
+ main (args : String[]) : void
- passMonth () : void
- getMatcher (input : String, regex : String) : Matcher
Account
- allAccounts : ArrayList<Account>
- bank : Bank
- id : int
- money : int
- remainingDuration : int
- interest : int
- customer : Customer
+ Account (bank : Bank, customer : Customer, id : int, money : int, duration : int, interest : int) «constructor»
+ passMonth () : void
+ deleteAccount (account : Account)
+ getId () : int
+ getAmountOfMoneyForLeaving () : double
+ getBank () : Bank
- passMonthEach () : void
Bank
- allBanks : ArrayList<Bank>
- name : String
+ Bank (name : String) «constructor»
+ getBankWithName (name : String) : Bank
+ isThereBankWithName (name : String) : boolean
+ getAccountInterestFromName (type : String) : int
+ getInterest (type : int) : int
+ getName () : String
Loan
- allLoans : ArrayList<Loan>
- customer : Customer
- deuration : int
- remainingPayments : int
- interest : int
- amount : int
+ Loan (customer : Customer, duration : int, interest : int, amount : int) «constructor»
+ passMonth () : void
- getPaymentAmount () : double
- passMonthEach () : void
Customer
- allCustomers : ArrayList<Customer>
- name : String
- moneyInSafe : double
- allActiveAccounts : ArrayList<Account>
- totalNumberOfAccountsCreated : int
- negativeScore : int
+ Customer (name : String, moneyInSafe : double) «constructor»
+ getCustomerByName (name : String) : Customer
+ getName () : String
+ createNewAccount (bank : Bank, money : int, duration : int, type : int) : void
+ leaveAccount (accountId : int) : void
+ canPayLoan (amount : double) : boolean
+ getMoneyInSafe () : double
+ setMoneyInSafe (moneyInSafe : double) : void
+ getLoan (duration : int, interest : int, money : int) : void
+ payLoan (amount : double) : void
+ canGetLoan () : boolean
+ getNegativeScore () : int
+ addNegativeScore () : void
+ hasActiveAccountInBack (bank : Bank) : boolean
- getAccountWithId (id : int) : Account

۲ شطرنج‌نامه

ایکس و ایگرگ دو شطرنج‌باز حرفه‌ای هستند که متأسفانه در جزیره‌ای دور افتاده به همراه شما و البته یک عدد لپ تاپ (و تعدادی باطری اضافی برای لپ تاپ!) گیرافتاده اند و از آن جایی که در جزیره غذایی نیست، تصمیم دارند شما را به عنوان غذا بخورند. با این وجود از آن جایی که آن‌ها خیلی عاشق شطرنج هستند ولی هیچ مهره شطرنجی همراهشان ندارند و از طرفی هم می‌دانند که شما برنامه نویسی بلدید، به شما قول داده‌اند که اگر بتوانید کد یک بازی شطرنج را برای آن‌ها بزنید تا شطرنج‌بازی کنند، از خیر خوردن شما می‌گذرند و آن قدر بازی خواهند کرد تا خودشان از گرسنگی بمیرند! البته آن‌ها برای این که کارشان سریع‌تر راه بیفتد، یکسری قوانین شطرنج را هم بی‌خیال شده‌اند و صرفاً اگر مواردی که آن‌ها از شما می‌خواهند را پیاده سازی بکنید، از خیر خوردن شما خواهند گذشت.

همان طور که از بند بالا مشخص است، هدف شما در این سؤال پیاده سازی بازی شطرنج خواهد بود. قوانین این بازی شطرنج تا حد زیادی مشابه قوانین شطرنج واقعی است و صرفاً چند قانون آن برای سادگی حذف شده‌اند و یک مورد ویژگی متفاوت به آن اضافه شده است.

- در این نوع شطرنج چیزی به نام کیش و مات نداریم و صرفاً با **زدن شاه**، بازی به پایان می‌رسد. یعنی نیازی به انجام هیچ گونه چک برای کیش شدن و یا مات شدن نیست و خود بازیکنان اگر حواسشان نباشد و شاهشان در وضعیت تهدید باشد و زده شود، بازنده خواهند بود. پس از این لحاظ شما نیاز به بررسی وضعیت کیش و مات ندارید و فقط باید زده نشدن شاه بررسی شود.
- قلعه کردن و حرکت آن پاسان در بازی وجود ندارند و نیازی به پیاده سازی آن‌ها نیست. (پیاده سازی آن‌ها نمره امتیازی هم ندارد.)
- این موضوع که سربازها با رسیدن به آخر زمین تبدیل به وزیر و... بشوند وجود ندارد.
- در ابتدای شروع بازی تعداد مشخصی حرکت به عنوان لیمیت تعیین خواهد شد. در صورتی که این محدودیت به اتمام برسد (یعنی دقیقاً به همان اندازه در مجموع دو نفر حرکت انجام داده باشند) و هیچ کسی بازی را برنده نشده باشد بازی مساوی اعلام می‌شود. مساوی به شکل رایج شطرنج (پات) که قوانین مختلفی نظیر تعداد و نوع مهره‌ها و حرکات ممکن و... باید در آن چک شود، در این جا وجود ندارد.

- امکان unselect کردن مهره انتخابی برای حرکت وجود دارد و قانونی که در بین شطرنج‌بازان معروف به "دست به مهره، حرکت" است وجود ندارد.
- قابلیت undo کردن حرکات وجود دارد (در این مورد توضیح بیش‌تری خواهیم داد).

قوانین کلی شطرنجی که باید پیاده سازی کنید:

شروع حرکت با مهره سفید است.

در صورت زده شدن شاه در این شطرنجی که قرار است پیاده سازی کنید، بازی به پایان رسیده و فردی که شاه را از دست داده بازی را می‌بازد. مهم‌ترین تفاوت این شطرنج با شطرنج واقعی هم همین مورد و نبودن وضعیت کیش است.

سربازها در ابتدا می‌توانند یک یا دو خانه به جلو بروند و پس از آن تنها یک خانه به جلو می‌روند. نحوه زدن مهره‌های حریف توسط سرباز به صورت ضربدری است. در حین حرکت در جلوی سرباز نباید هیچ مهره‌ای باشد.

فیل به صورت مورب حرکت می‌کند و مهره‌های دیگر را هم به همین صورت می‌زند. در حین حرکت در مسیر حرکت مورب فیل نباید هیچ مهره‌ای باشد.

حرکت رخ به صورت افقی و عمودی است و مهره‌های دیگر را هم به همین صورت می‌زند. در حین حرکت در مسیر حرکت رخ نباید هیچ مهره‌ای باشد.

حرکت وزیر مانند ترکیب حرکت رخ و فیل است و هم مورب و هم عمودی و افقی حرکت می‌کند مهره‌های دیگر را هم به همین صورت می‌زند. در حین حرکت در مسیر حرکت وزیر نباید هیچ مهره‌ای باشد.

حرکت اسب به صورت L شکل است. یعنی دو خانه به جلو و یک خانه به راست یا دو خانه به جلو و یک خانه به چپ یا دو خانه به راست و یک خانه بالا و... و مهره‌های دیگر را هم به همین صورت می‌زند. در حین حرکت می‌تواند از روی سایر مهره‌ها بپرد و قرار داشتن مهره‌های دیگر در سر راه اسب مشکلی برای حرکت ایجاد نمی‌کند.

البته برنامه‌ای که باید بنویسید، صرفاً خود بازی شطرنج نیست. بلکه باید مکانیزم حساب کاربری و ثبت نام و لاگین و... را مطابق مواردی که در ادامه برای شما گفته می‌شود، پیاده سازی کنید.

در این برنامه، در اصل سه منو (Menu) داریم. یکی منوی ابتدای بازی یا منوی ثبت نام است. یکی منوی اصلی و یکی هم منوی خود بازی. در ادامه به توضیح دستوراتی که در هر کدام از این منوها وارد می‌شود می‌پردازیم. توجه کنید که در هنگام شروع برنامه، به طور پیش فرض، کاربر در منوی ثبت نام قرار دارد. در تمامی منوها دستور help وجود دارد که

توضیح آن را در هر منو قرار داده‌ایم. ضمناً توجه کنید تمامی پیغام‌های انجام موفقیت‌آمیز دستورات یا خطاها، در یک خط مجزا چاپ می‌شوند و بعد به خط بعدی می‌رود.
همچنین توجه کنید که نماد [] در دستورات پایین در خود دستور نوشته نمی‌شوند. صرفاً برای نمایش این که در این قسمت عبارتی از سمت کاربر باید وارد شود نوشته شده‌اند.
یعنی مثلاً یک عبارت معتبر برای

```
register [username] [password]
```

به صورت:

```
register ali 1234
```

می‌تواند باشد.

منوی ثبت نام:

در ابتدای شروع برنامه کاربر در این منو قرار دارد.
دستورات:

```
register [username] [password]
```

همان طور که مشخص است، یک کاربر با نام کاربری و پسورد مشخص شده را ایجاد می‌کند. نام کاربری و پسورد باید فقط شامل حروف الفبای انگلیسی، اعداد و کاراکتر آندرلاین _ باشند. خطاهای مربوط به این دستور به این ترتیب چک می‌شوند و هر خطا که رخ داده بود، پیغام همان خطا چاپ شده و سایر خطاها بررسی نمی‌شوند. اگر هیچ خطایی رخ نداد و عملیات موفقیت‌آمیز بود، پیغام

```
register successful
```

چاپ خواهد شد.
خطاها:
اگر نام کاربری شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
username format is invalid
```

اگر پسورد شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
password format is invalid
```

اگر کاربری با username گفته شده از قبل وجود داشت پیغام:

```
a user exists with this username
```

چاپ خواهد شد.

```
login [username] [password]
```

برای لاگین کردن به حساب کاربری مشخص با نام کاربری و پسورد داده شده استفاده می‌شود. نام کاربری و پسورد باید فقط شامل حروف الفبای انگلیسی، اعداد و کاراکتر آندرلاین _ باشند. خطاهای مربوط به این دستور به این ترتیب چک می‌شوند و هر خطا که رخ داده بود، پیغام همان خطا چاپ شده و سایر خطاها بررسی نمی‌شوند. اگر هیچ خطایی رخ نداد و عملیات موفقیت‌آمیز بود، پیغام

```
login successful
```

چاپ خواهد شد و پس از آن کاربر به طور خودکار وارد منوی اصلی خواهد شد.
خطاها:
اگر نام کاربری شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
username format is invalid
```

اگر پسورد شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
password format is invalid
```

اگر کاربری با username گفته شده وجود نداشت:

```
no user exists with this username
```

اگر پسورد غلط بود:

```
incorrect password
```

چاپ خواهد شد.

```
remove [username] [password]
```

برای حذف کردن یک حساب کاربری مشخص با نام کاربری و پسورد داده شده استفاده می‌شود. نام کاربری و پسورد باید فقط شامل حروف الفبای انگلیسی، اعداد و کاراکتر آندرلاین _ باشند. خطاهای مربوط به این دستور به این ترتیب چک می‌شوند و هر خطا که رخ داده بود، پیغام همان خطا چاپ شده و سایر خطاها بررسی نمی‌شوند. اگر همه موفق آمیز بودند پیغام

```
removed [username] successfully
```

چاپ خواهد شد که به جای username باید نام کاربری فرد حذف شده نشان داده شود.

خطاها:

اگر نام کاربری شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
username format is invalid
```

اگر پسورد شامل کاراکترهایی به جز کاراکترهای ذکر شده بود، پیغام:

```
password format is invalid
```

اگر کاربری با username گفته شده وجود نداشت:

```
no user exists with this username
```

اگر پسورد غلط بود:

```
incorrect password
```

چاپ خواهد شد.

```
list_users
```

این دستور لیست تمامی کاربرانی که وجود دارند را به ترتیب الفبایی (Lexicograph - ical) نمایش می‌دهد.

```
help
```

این فرمان انواع دستوراتی که در این بخش قابل نمایش هستند را نمایش می‌دهد. از این دستور صرفاً برای اطمینان حاصل کردن از این که در منوی درست قرار دارید. خروجی این دستور در این منو به صورت زیر است:

```
register [username] [password]
login [username] [password]
remove [username] [password]
list_users
help
exit
```

```
exit
```

بیانگر اتمام اجرای برنامه است و بعد از چاپ پیام

```
program ended
```

اجرای برنامه پایان می‌پذیرد.

منوی اصلی:
همان طور که گفته شد، در صورت موفقیت‌آمیز بودن login وارد این منو خواهید شد.
دستورات:

```
new_game [username] [limit]
```

“

با این دستور یک بازی جدید شروع می‌شود. در این بخش باید username یک کاربر دیگر وارد شود و بدین ترتیب بازی با آن کاربر شروع خواهد شد. کاربری که اکنون لاگین کرده و دستور را زده است در حین بازی به عنوان مهره سفید و بازیکنی که در این دستور نام او وارد شده به عنوان مهره سیاه خواهد بود. توجه کنید که در این برنامه قرار نیست چیزی نظیر لاگین شدن همزمان دو کاربر و... را هندل کنید و همه کارها در یک برنامه

و توسط یک کاربر انجام می‌شود. limit هم یک عدد است و بیانگر محدودیت تعداد حرکات (نوبت‌ها در بازی) است. اگر این عدد ۰ باشد به معنی نبودن هیچ محدودیتی در بازی است. جزئیات بیش‌تر مربوط به limit در بخش مربوط به بازی توضیح داده شده‌اند.
خطاهای این دستور:
اگر username از کاراکترهایی که در منوی لاگین توضیح دادیم تشکیل نشده بود، پیام:

```
username format is invalid
```

اگر limit عددی کوچک‌تر از ۰ بود:

```
number should be positive to have a limit or 0 for no limit
```

اگر کاربری که دستور را زده است، نام کاربری خودش را وارد کرد:

```
you must choose another player to start a game
```

اگر کاربری با این نام کاربری وجود نداشت:

```
no user exists with this username
```

چاپ خواهد شد.
در صورت اجرای موفقیت‌آمیز دستور، پیام:

```
new game started successfully between [first] and [second] with limit [limit]
```

جاپ می‌شود که در آن first نام کاربری بازیکن سفید و second نام کاربری بازیکن سیاه و limit هم عدد محدودیت بازی است (در حالتی که ۰ وارد شده بود به معنی عدم محدودیت هم عدد ۰ نمایش داده خواهد شد و استثنایی وجود ندارد.)

```
scoreboard
```

پیش از توضیح خروجی این دستور باید در مورد امتیاز دهی در بازی صحبت بکنیم. هر برد معمولی (از طریق زدن شاه) ۳ امتیاز و باخت به این شکل ۰ امتیاز دارد. در صورتی که یکی از بازیکنان انصراف بدهد (با دستوری که در بخش مربوط به بازی توضیح می‌دهیم)،

برنده بازی ۲ امتیاز دریافت کرده و شخصی که انصراف داده ۱ امتیاز منفی کسب می‌کند. در صورت تساوی بازی (که در اثر اتمام limit اتفاق می‌افتد) هر بازیکن ۱ امتیاز می‌گیرد.
بعد از اجرای این دستور باید کاربران با این فرمت نوشته بشوند:

```
[username] [score] [wins] [draws] [losses]
```

منظور از wins و draws و losses تعداد بردها، تساوی‌ها و باخت هاست.
ترتیب مرتب سازی کاربران هم به ترتیب از بالاترین اولویت به کمترین به این صورت است: بیش‌ترین امتیاز در صورت برابری امتیاز بیش‌ترین تعداد برد در صورت برابری تعداد برد بیش‌ترین تعداد تساوی در صورت برابری تساوی کمترین باخت و در صورت برابری تمامی موارد، براساس حروف الفبا (lexicographical) به طور صعودی (یعنی a زودتر از z می‌آید و...)

```
list_users
```

کاملاً مشابه همین دستور که در منوی قبلی توضیح داده شد، عمل می‌کند.

```
help
```

مشابه همان چیزی است که در بخش قبل توضیح دادیم. خروجی آن عیناً به این شکل است:

```
new_game [username] [limit]
scoreboard
list_users
help
logout
```

```
logout
```

با این دستور کاربر از حساب کاربری خود خارج می‌شود. پس از اجرای این دستور پیام

```
logout successful
```

چاپ شده و کاربر وارد منوی ثبت نام که پیش‌تر توضیح دادیم، می‌شود.

منوی بازی:

در این بخش علاوه بر این که به دستورات اشاره می‌کنیم، به موارد خاصی که باید در منطق به درستی پیاده سازی شوند هم اشاره می‌شود.

به عنوان اولین نکته به تصویر زیر که از صفحه شطرنج است توجه کنید:

فایل تصویر به طور جداگانه باکیفیت بالاتر قرار گرفته است.

نحوه قرار گیری مهره‌ها در صفحه بازی به همین شکل خواهد بود.

در این جا اولین دستور مطرح می‌شود:

```
select [x],[y]
```

با این دستور یک مهره را انتخاب می‌کنیم. توجه کنید که برای اشاره به یک خانه در این جا به جای حروف A تا H هم از اعداد استفاده خواهیم کرد. یعنی مثلاً در ابتدای بازی، وزیر سفید به صورت select ۱،۴ و شاه سیاه با select ۵،۸ انتخاب می‌شود.

خطاهای این دستور:

اگر مختصات به طور کلی از محدوده خارج بود (یعنی هر مؤلفه در بازه ۱ تا ۸ قرار نمی‌گرفتند):

```
wrong coordination
```

اگر مهره قرار گرفته در خانه مشخص شده مربوط به حریف بود:

```
you can only select one of your pieces
```

اگر هیچ مهره‌ای در خانه نبود:

```
no piece on this spot
```

چاپ می‌شوند.
اگر هم مهره خودی در آن موقعیت بود، پیغام:

```
selected
```

چاپ می‌شود. توجه کنید که اگر یک مهره از قبل انتخاب شده باشد و بخواهیم مهره جدیدی را انتخاب کنیم، صرفاً با نوشتن دوباره دستور برای مهره جدید، مهره جدید انتخاب خواهد شد و دوباره همان خروجی

```
selected
```

را خواهیم داشت. حتی با انتخاب دوباره مهره انتخاب شده هم با همین نتیجه رو به رو خواهیم بود.

```
deselect
```

این دستور باعث می‌شود مهره‌ای که در وضعیت انتخاب شده بوده، از وضعیت انتخاب شده خارج شود.
تنها خطای این دستور مربوط به وقتی است که هیچ مهره‌ای انتخاب نشده باشد و در آن صورت عبارت:

```
no piece is selected
```

چاپ می‌شود.
در صورتی هم که به درستی مهره از حالت انتخاب خارج شود، پیغام

```
deselected
```

چاپ خواهد شد.

```
move [x],[y]
```

این حرکت همان طور که واضح است، باعث حرکت مهره به نقطه مشخص شده می‌شود.
خطاهای این دستور:
اگر قبل از این حرکتی در این نوبت انجام شده بود:

```
already moved
```

اگر مختصات به طور کلی از محدوده خارج بود (یعنی هر مؤلفه در بازه ۱ تا ۸ قرار نمی‌گرفتند):

```
wrong coordination
```

اگر هیچ مهره‌ای انتخاب نشده بود:

```
do not have any selected piece
```

اگر خانه مورد نظر توسط مهره‌های خودی از قبل پر بود یا مانعی در مسیر برای حرکت به آن نقطه وجود داشت (مطابق قوانین شطرنج که در ابتدای داک توضیح داده شده‌اند):

```
cannot move to the spot
```

چاپ خواهد شد.
در صورت اجرای موفقیت‌آمیز، اگر حرکت به درستی انجام شد و در خانه مد نظر مهره حریف قرار داشت و زده شد عبارت:

```
rival piece destroyed
```

و اگر صرفاً حرکت انجام شد و مهره‌ای زده نشد عبارت:

```
moved
```

چاپ خواهد شد.

```
next_turn
```

نوبت را به حریف منتقل می‌کند و در صورت اجرای موفقت آمیز پیغام:

```
turn completed
```

چاپ می‌شود. توجه کنید که طبق قوانین شطرنج هر نفر باید در نوبت خود حتماً یک حرکت انجام بدهد. بنابراین اگر هیچ حرکتی انجام نشده بود، پیغام خطای:

```
you must move then proceed to next turn
```

نوشته شود.

```
show_turn
```

این دستور برای مشخص شدن این است که نوبت کدام بازیکن است. خروجی آن به فرم زیر است:

```
it is player [username] turn with color [white
black]
```

```
undo
```

یکی از ویژگی‌های منحصر به فرد شطرنجی که باید پیاده سازی کنید، قابلیت undo است. هر بازیکن در کل طول یک بازی می‌تواند دو بار از این قابلیت و در هر نوبت حداکثر یک بار استفاده کند. این قابلیت بدین صورت است که اگر بازیکنی حرکتی انجام داده باشد، در صورتی که بخواهد و طبق شرایط بالا تعداد دفعات استفاده‌اش تمام نشده باشد، حرکت خود را برگرداند. توجه کنید که چون تغییر نوبت با next_turn انجام می‌شود، همچنان بعد از حرکت نوبت با بازیکن فعلی است تا زمانی که دستور next_turn زده شود.
خطاهای این دستور:
اگر پیش‌تر به اندازه تعداد کل undo های مجاز (یعنی ۲) بار از این قابلیت استفاده شده باشد.

```
you cannot undo anymore
```

اگر بازیکن در این نوبت حرکتی انجام نداده باشد که بخواهد undo کند:

```
you must move before undo
```

اگر در همین نوبت از undo استفاده کرده باشد:

```
you have used your undo for this turn
```

چاپ خواهند شد.
در صورت اجرای درست دستور نیز باید عبارت:

```
undo completed
```

چاپ بشود.
توجه کنید که بعد از اجرای ،undo مهره‌ای که از قبل select شده همچنان در همین حالت باقی می‌ماند.

```
undo_number
```

تعداد دفعات باقی مانده undo یک بازیکن را نشان می‌دهد. در ابتدای بازی این عدد دو است و با انجام حرکت undo کاهش می‌یابد تا در نهایت بعد از دو بار انجام این حرکت، به صفر می‌رسد. فرم پیام خروجی به صورت:

```
you have [n] undo moves
```

خواهد بود که n تعداد undo های باقی مانده است.

برای دستورات بعدی باید توضیحی در مورد نحوه نمایش نام هر مهره در بازی بدهیم.
نام مهره‌ها در بازی به این صورت است که ابتدا حرف اختصاری نوع مهره با حروف بزرگ و سپس رنگ مهره با حروف کوچک آورده می‌شود.
حرف اختصاری مهره‌ها بدین شرح است: سرباز P ـ رخ R ـ اسب N ـ فیل B ـ وزیر Q و شاه .K به عنوان مثال شاه سفید با علامت Kw شناخته می‌شود.
حال به توضیح خود دستور می‌پردازیم:

```
show_moves
```

این دستور کل حرکاتی را که بازیکن فعلی انجام داده است، به ترتیب با این فرمت نمایش می‌دهد: (منظور از بازیکن فعلی، بازیکنی است که در هنگام زده شدن دستور نوبت اوست)
اگر در طول فرآیند حرکت مهره‌ای زده نشده باشد:

```
[Name of Piece that moved] [x],[y] to [x],[y]
```

اگر مهره‌ای زده شده باشد:

```
[Name of Piece that moved] [x],[y] to [x],[y] destroyed [Name of piece destroyed]
```

به عنوان مثال اگر سرباز سفید با حرکت از ردیف ۴ و ستون ۱ به ردیف ۵ و ستون ۲ رفته باشد و در این فرآیند سرباز سیاه را از بین برده باشد، حرکت به صورت

```
Pw 4,1 to 5,2 destroyed Pb
```

نمایش داده خواهد شد.

```
show_moves -all
```

این دستور مشابه دستور قبلی است با این تفاوت که حرکات تمامی بازیکنان را به ترتیب از ابتدای شروع بازی نمایش می‌دهد.

```
show_killed
```

این دستور مهره‌های زده شده بازیکن فعلی را نشان می‌دهد. (یعنی مهره‌هایی از او که حریف آن‌ها را زده است).
فرمت نمایش آن‌ها به این صورت است:

```
[Name of Piece] killed in spot [x],[y]
```

ترتیب نمایش مهره‌ها، به ترتیب زمان زده شدن است. (مهره‌ای که زودتر زده شده، زودتر نمایش داده می‌شود)

```
show_killed -all
```

مشابه دستور بالاست ولی همه مهره‌های زده شده در بازی را به ترتیب زده شدن نمایش می‌دهد.

```
show_board
```

این دستور صفحه شطرنج را به همراه مهره‌های حاضر در آن نمایش می‌دهد. برای مثال در ابتدای بازی فرمت نمایش صفحه به صورت زیر است و به طور کلی باید به همین شکل نمایش بدهید.

```
Rb|Nb|Bb|Qb|Kb|Bb|Nb|Rb|
Pb|Pb|Pb|Pb|Pb|Pb|Pb|Pb|
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |
Pw|Pw|Pw|Pw|Pw|Pw|Pw|Pw|
Rw|Nw|Bw|Qw|Kw|Bw|Nw|Rw|
```

توجه کنید که خانه‌های خالی با دو اسپیس نمایش داده می‌شوند. در صورتی که می‌خواهید عیناً ساختار بالا را کپی کنید، به صفحه کوئرا مراجعه کنید.

یعنی زاویه دید، مشابه عکسی است که بالاتر از بازی شطرنج نمایش دادیم و سفید در پایین تصویر و سیاه در بالای تصویر خواهد بود.

```
help
```

مشابه دستور help در سایر بخش‌هاست. در این جا باید عیناً این خروجی نمایش داده شود:

```
select [x],[y]
deselect
move [x],[y]
next_turn
show_turn
undo
undo_number
show_moves [-all]
show_killed [-all]
show_board
help
forfeit
```

```
forfeit
```

این دستور مربوط به انصراف از بازی می‌شود و بازیکنی که این دستور را وارد کند، بازنده بازی خواهد بود و نفر دیگر برنده بازی خواهد بود. همان طور که بالاتر گفتیم، با این اتفاق، حریف که برنده بازی شده، ۲ امتیاز می‌گیرد و کسی که انصراف داده منفی ۱ امتیاز دریافت می‌کند (یعنی ۱ امتیاز از او کم می‌شود).

بعد از اجرای این دستور باید این دو پیام در دو خط پشت سرهم نمایش داده شوند:

```
you have forfeited
```

```
player [username] with color [blackwhite] won
```

و به جای username و black/white هم نام کاربری و رنگ بازیکن برنده قرار می‌گیرد. پس از آن هم کاربر به منوی اصلی بازگردانده می‌شود.

بدین ترتیب توصیف دستورات به پایان رسید. صرفاً باید مواردی را در مورد پایان بازی ذکر کنیم.

در صورتی که شاه حریف زده شود، بازی به نفع شخصی که شاه حریف را زده است، تمام می‌شود. این موضوع که شاه حریف زده شده در دستور next_turn بررسی می‌شود.
در صورت اتمام بازی با برد یکی از طرفین پیامی به صورت:

```
player [username] with color [blackwhite] won
```

چاپ می‌شود.
در مورد شرایط تساوی، همان طور که گفته شد، تنها باید وضعیت limit تعداد حرکات را بررسی بکنید. با گذشت هر نوبت به نوعی یک واحد به limit مد نظر نزدیک‌تر می‌شویم. یعنی به نوعی اگر limit عدد ۲۰ باشد، هر یک از طرفین می‌تواند ۱۰ نوبت بازی کند. در این مثال، اگر بعد از گذشت ۲۰ نوبت در کل، هیچ یک از طرفین برنده بازی نشد، بازی با تساوی به اتمام می‌رسد. توجه کنید که اگر limit برابر ۰ باشد، یعنی هیچ محدودیتی نیست و بازی کلاً با تساوی به اتمام نخواهد رسید. در صورت تساوی عبارت زیر چاپ می‌شود:

```
draw
```

و کاربر به منو اصلی باز گردانده می‌شود.
همان طور که پیش‌تر هم گفتیم، در اثر تساوی به هر کدام از بازیکنان ۱ امتیاز داده می‌شود.
با توجه به توضیحات بالا نیازی به چک کردن شرایط واقعی تساوی در شطرنج ندارید.
در نهایت توجه کنید که اگر در هر کدام از منوها، دستوری زده شود که با دستورات آن منو تطابق نداشته باشد، باید پیام

```
invalid command
```

در یک خط چاپ شود و به خط بعدی برویم.
در مورد خطاهای دستورات، توجه کنید که ترتیب چک شدن آن‌ها به ترتیبی است که در این داک نوشته شده‌اند و در صورت رخ دادن اولین خطا و چاپ پیام خطا، سایر خطاها بررسی نمی‌شوند.

۳ نصّاب پنجره

وینیسیوس جونیور که در پی ویروس کرونا خانه نشین شده است تصمیم گرفته تا سیستم عامل لپ تاپ خود را تغییر دهد؛ اما چون تجربه‌ای در این کار ندارد از شما کمک خواسته است. در این سؤال شما بایستی دستوراتی که او به شما می‌دهد را پیاده کنید. در این سؤال ابتدا با نصب سیستم عامل و ساخت درایو ها شروع کرده سپس به سراغ ساخت فایل و فولدر و کارهای مربوط به آن‌ها خواهیم رفت؛ اما ابتدا یکسری نکات کلی در مورد این سؤال:

در مورد فایل:

در این سؤال با ۳ مدل فایل سروکار داریم عکس (فرمت img)، ویدئو (فرمت $mp4$) و متن (فرمت txt) هستند.

یک سری ویژگی‌ها در تمام این نوع فایل‌ها وجود دارند مانند آدرس و اسم و .... . که در ادامه با دستورات مربوط به آن‌ها آشنا می‌شوید. اما بعضی ویژگی‌ها هستند که بنا بر نوع فایل متفاوت‌اند. این ویژگی‌ها به ترتیب برای هر فایل به شکل زیر هستند:

ویدیو:

Quality(240p, 360p,720p,1080p,2160p)

Video Length

(که به فرمت hh:mm:ss داده می‌شه)

متن:

Text

(نگران نباشین قرار نیست با فایل متنی واقعی کار کنین صرفاً کافیه یه رشته در نظر بگیرین براش)

عکس:

Resolution

(به صورت دو عدد با علامت * در بینشون مثل $2048 * 1536$)

Extension(jpg, png)

در مورد فولدر:

هر فولدر از تعدادی فایل و فولدر دیگر تشکیل شده است.

نکته مهم: توجه کنید که در هر بخش اگر دستور ورودی طبق فرمت داده شده نبود کافی است دستور خطای گفته شده را چاپ کنید و برنامه در همان بخش باقی می‌ماند.

نکته مهم ۲: دقت کنید که اروهایی که برای هر دستور در ادامه گفته شده را به همین ترتیب داک، چک کنید

نکته مهم ۳: نام فایل‌ها و فولدرها sensitive case نیست و در واقع دو فولدر a و A نمی‌توانند در یک مکان باشند (برای دو فایل نیز به همین شکل)(در ادامه درباره این ارور توضیح داده شده)

نکته مهم ۴: با توجه به نکته ۳ در نظر داشته باشید که هر جایی که نیاز است تا اسم یک فایل و فولدر (و نه درایو) را بررسی کنید مهم نیست که حروف آن اسم بزرگ باشد یا کوچک، به عنوان مثال اگر در فولدری هستید که فایلی به نام $AdvaNCe$ دارید اگر دستور $delete\ file\ aDVanCE$ وارد شود، دستور معتبری است (در ادامه توضیح داده شده است)

نکته مهم ۵: از space های اضافی در ابتدا و انتهای دستورات صرف نظر کنید یعنی دستور end نیز صحیح تلقی می‌شود.(این دستور را در انتهای توضیحات سؤال می‌توانید ببینید)

خب بریم سراغ دستورات (به کوچکی و بزرگی حروف دستورات و ارور های که باید چاپ کنید، دقت کنید):

در ابتدا دستوری برای نصب سیستم عامل به فرمت زیر وارد می‌شود (ورژن سیستم عامل می‌تواند شامل کاراکترهای غیر عدد نیز باشد ولی شامل space نیست) این دستور تضمین می‌شود که فقط یک بار وارد شود.(البته الزاماً دستور اول برنامه، این دستور نیست و باید معتبر بودن آن را چک کنید)

```
install OS #name #version
```

سپس دو عدد در یک خط با یک فاصله وارد می‌شوند که به ترتیب (از چپ) نشان دهنده حجم هارد و تعداد درایوهای سیستم هستند (حجم هارد را در مقیاس MB در نظر بگیرید و دقیقاً برابر با عدد داده شده (از توان ۲ نبودن آن چشم پوشی کنید)) این دستور تضمین می‌شود که فقط یک بار وارد شود.(البته الزاماً دستوری که بلافاصله بعد دستور بالا وارد می‌شود، این دستور نیست و باید معتبر بودن آن را چک کنید)

```
#hardSize #drivesNum
```

سپس در #drivesNum خط در هر خط یک حرف بزرگ انگلیسی داده می‌شود ($A-Z$) که نام هر درایو را نشان می‌دهد سپس با یک فاصله حجم مورد نظر برای آن درایو وارد می‌شود.(فقط عدد و بدون MB است و حتی می‌تواند صفر باشد) توجه کنید که نباید نام درایو ها تکراری باشند.

توجه: اولین درایوی که وارد می‌شود و معتبر باشد، به عنوان درایو سیستم در نظر گرفته می‌شود که به این معنی است که در ابتدای برنامه پس از ورود به محیط سیستم عامل، در داخل آن درایو قرار داریم

```
#driveName #driveSize
```

اگر نام وارد شده تکراری بود یا بیش از یک حرف داشت یا شامل کاراکترهای دیگری غیر از حروف بزرگ بود باید ارور زیر چاپ شود:

```
invalid name
```

اگر حجم درایو داده شده + مجموع حجم درایو های ثبت شده قبلی از حجم هارد بیشتر بود ارور زیر چاپ شود:

```
insufficient hard size
```

تضمین می‌شود که مجموع اندازه درایو ها از اندازه هارد کمتر نخواهد بود (اگر بیشتر باشد ارور بالا را باید چاپ کنید اگر مساوی بود برنامه ادامه می‌یابد)
حال که سیستم عامل با موفقیت نصب شد وقتشه که شروع کنیم به ساخت فایل و فولدر و کار کردن با اون‌ها

```
open #folderName
```

با این دستور وارد فولدر با اسم داده شده می‌شوید (در صورت معتبر بودن دستور، باز کردن فولدر به این معنی است که مکان فعلی تغییر می‌کند و وارد آن فولدر می‌شوید)
توجه: این دستور برای فایل‌ها وجود ندارد و حتی در صورتی که فایل با اسم داده شده نیز وجود داشته باشد اجرا نمی‌شود.
اگر فولدری با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود

```
invalid name
```

```
go to drive #driveName
```

این دستور در هر کجای برنامه وارد شود بایستی وارد درایو با نام داده شده شوید به این معنی که مکان فعلی تغییر می‌کند و وارد آن درایو می‌شوید.
اگر درایوی با نام داده شده وجود نداشت باید ارور زیر چاپ شود

```
invalid name
```

```
back
```

با این دستور یک مرحله به عقب برمی‌گردید.
مثال:
اگر مکان فعلی

```
H:\APSpirng98\HW3
```

باشد و دستور back وارد شود، مکان به

```
H:\APSpirng98
```

تغییر می‌کند
مثال ۲: اگر مکان فعلی باشد\:H و دستور back وارد شود، مکان شما تغییر نمی‌کند و اروری نیز چاپ نمی‌شود.

```
create folder #name
```

با دستور بالا فولدری با اسم داده شده در مکان فعلی ساخته می‌شود. نام فولدر می‌تواند شامل کاراکترهای غیر حروف نیز باشد
اگر فولدر دیگری با آن اسم در آن مکان وجود داشت ارور زیر چاپ شود:

```
folder exists with this name
```

اگر دستور معتبر بود پس از ساخت فولدر عبارت زیر چاپ شود:

```
folder created
```

```
create file #name #format #size
```

با دستور بالا فایلی با اسم داده شده در مکان فعلی ساخته می‌شود. سایز داده شده برحسب MB است. نام فایل می‌تواند شامل کاراکترهای غیر حروف نیز باشد.
اگر فایل دیگری با آن اسم در آن مکان وجود داشت ارور زیر چاپ شود:

```
file exists with this name
```

اگر فرمت داده شده از ۳ نوع $mp4$، img، txt نبود باید ارور زیر چاپ شود:

```
invalid format
```

اگر حجم فایل داده شده + حجم فایل‌های داخل آن درایو از حجم آن درایو بیشتر بود (فضای کافی برای ساخت فایل وجود نداشت) باید ارور زیر چاپ شود:

```
insufficient drive size
```

توجه کنید که format معتبر داده شده یکی از ۳ نوع $mp4$،img، txt است که بسته به اینکه کدام نوع باشد باید فایلی از نوع متن، عکس یا ویدیو بسازید.

سپس اگر ورودی مشکلی نداشت، به نسبت فرمت فایل، معیارها را دریافت می‌کند. به این صورت که به ترتیب آورده شده در بالا، معیارها چاپ می‌شود و مقادیر مورد نظر از ورودی دریافت می‌شود.

مثلاً اگر ویدیو بود باید چاپ کنید:

```
Quality:
```

سپس ورودی بگیرید و سپس چاپ کنید:

```
Video Length:
```

و ورودی بگیرید.(تضمین می‌شود که این ورودی‌ها معتبر باشند)

در نهایت اگر دستور معتبر بود پس از گرفتن ویژگی‌های فایل که در بالا گفته شد، و پس از ساخت فایل عبارت زیر چاپ شود:

```
file created
```

```
delete file #fileName
```

با این دستور فایل با اسم داده را حذف می‌کنید و حجم آن را نیز از حجم استفاده شده از فضای درایو حذف می‌کنید.

اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر دستور معتبر بود پس از حذف فایل عبارت زیر چاپ شود:

```
file deleted
```

```
delete folder #folderName
```

با این دستور فولدر با اسم داده را حذف می‌کنید. دقت کنید که با حذف فولدر تمام فایل‌ها و فولدرهای داخل آن نیز حذف خواهند شد.
اگر فولدری با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر دستور معتبر بود پس از حذف فولدر عبارت زیر چاپ شود:

```
folder deleted
```

```
rename file #fileName #newName
```

اسم فایل با اسم داده شده را تغییر می‌دهید. اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر فایل دیگری به نام #newName در آن مکان وجود داشت باید ارور زیر چاپ شود:

```
file exists with this name
```

اگر دستور معتبر بود پس از تغییر نام فایل عبارت زیر چاپ شود:

```
file renamed
```

```
rename folder #folderName #newName
```

اسم فولدر با اسم داده شده را تغییر می‌دهید. اگر فولدر با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر فولدر دیگری به نام #newName در آن مکان وجود داشت باید ارور زیر چاپ شود:

```
folder exists with this name
```

اگر دستور معتبر بود پس از تغییر نام فولدر عبارت زیر چاپ شود:

```
folder renamed
```

```
status
```

این دستور اگر هنگامی که داخل درایو یا فولدری هستیم وارد شود ابتدا در یک خط آدرس مکان فعلی را چاپ کنید به عنوان مثال:

```
H:\Java Projects\APSpirng98\HW3
```

سپس نام تمام فایل‌ها و فولدرهای آن مکان را به ترتیب نام و به فرمت زیر چاپ کنید (به ترتیب ابتدا فولدرها و سپس عکس‌ها، متن‌ها، ویدیوها):
هر خط شامل یک اسم، فاصله، نوع آن فایل یا فولدر، فاصله، حجم آن فایل یا فولدر به عنوان مثال:

```
Folders:
Q2 50MB
q1 15MB
Files:
x img 2MB
a txt 80MB
B mp4 400MB
```

```
print drives status
```

در صورت وارد شدن این دستور باید به فرمت زیر اطلاعات درایو ها را چاپ کنید: هر خط شامل اسم درایو، فاصله، حجم کلی آن درایو، فاصله، حجم اشغال شده آن درایو توجه: ترتیب درایو ها همان ترتیب ساخته شدن آن‌ها باید باشد.
به عنوان مثال:

```
C 75000MB 48000MB
H 100000MB 500MB
```

در مورد دستورات کپی، کات و پیست که در ادامه آمده است به این نکات توجه کنید:
نکته ۱: دقت کنید که نمی‌توان همزمان هم فایل کپی (و یا کات) کرد و هم فولدر

نکته ۲: توجه کنید که اگر قبلاً تعدادی فایل یا فولدر کپی (و یا کات) شده بود و سپس دستور دیگری مبنی بر کپی (و یا کات) تعداد دیگری فایل یا فولدر وارد شد و معتبر بود، فایل یا فولدرهای قبلی، دیگر کپی (و یا کات) شده به حساب نمی‌آیند.
نکته ۳: هنگامی که فایل یا فولدری را paste می‌کنیم و عملیات موفق باشد، اگر کپی شده بود در مکان قبلی خود نیز باقی می‌ماند اما اگر cut شده بود از مکان قبلی پاک می‌شود.
نکته ۴: پس از paste کردن (در صورت معتبر بودن) فایل و یا فولدرهای کپی (و یا کات) شده دیگر کپی (و یا کات) شده به حساب نمی‌آیند.
نکته ۵: توجه کنید امکان اینکه فولدری را کپی کنید سپس وارد همان فولدر شده و عملیات دیگر از جمله عملیات paste را انجام دهید، وجود دارد. این کار در صورتی که آن فولدر را کات کرده باشید ـ حتی اگر patse نکرده باشید ـ وجود ندارد. البته جنین موردی در مورد cut در تست کیس‌ها وجود ندارد.

```
copy file #fileName # fileName ...
```

فایل‌ها با اسم داده شده را کپی می‌کند (می‌تواند ۱ یا چند فایل باشد). توجه کنید که حتی اگر فقط برای یکی از فایل‌ها ارور زیر رخ دهد، عملیات کپی کلاً انجام نمی‌شود.
اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود (حتی اگر یکی از اسم‌ها وجود نداشت)
توجه کنید با اولین بار که به ارور اشاره شده برخورد می‌کنید عملیات آنجا متوقف می‌شود و نیازی نیست برای دیگر اسم‌های داده شده چک کنید که فایلی وجود دارد یا نه

```
invalid name
```

اگر دستور معتبر بود عبارت زیر چاپ شود:

```
files copied
```

```
copy folder #folderName #folderName ...
```

فولدرها با اسم داده شده را کپی می‌کند (می‌تواند ۱ یا چند فولدر باشد). توجه کنید که حتی اگر فقط برای یکی از فولدرها ارور زیر رخ دهد، عملیات کپی کلاً انجام نمی‌شود.
اگر فولدری با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود (حتی اگر یکی از اسم‌ها وجود نداشت)

توجه کنید با اولین بار که به ارور اشاره شده برخورد می‌کنید عملیات آنجا متوقف می‌شود و نیازی نیست برای دیگر اسم‌های داده شده چک کنید که فولدری وجود دارد یا نه

```
invalid name
```

اگر دستور معتبر بود عبارت زیر چاپ شود:

```
folders copied
```

```
paste
```

فایل یا فولدرهای کپی شده را در محل فعلی پیست می‌کند.(اگر هیچ فایل یا فولدری قبلاً کپی نشده بود کاری انجام نمی‌دهد و اروری نیز چاپ نمی‌شود) توجه کنید که حتی اگر فقط برای یکی از فایل یا فولدرها ارور های زیر رخ دهد، عملیات paste کلاً انجام نمی‌شود.

توجه کنید با اولین بار که به ارورهای اشاره شده برخورد می‌کنید عملیات آنجا متوقف می‌شود و نیازی نیست برای بقیه فایل یا فولدرها چک کنید.

اگر فایلی کپی (و یا کات) شده بود و در مکان فعلی فایلی با همان نام وجود داشت باید ارور زیر چاپ شود:

```
file exists with this name
```

اگر فولدری کپی (و یا کات) شده بود و در مکان فعلی فولدری با همان نام وجود داشت باید ارور زیر چاپ شود:

```
folder exists with this name
```

اگر در مکان فعلی حجم کافی برای انجام عملیات paste وجود نداشت ارور زیر چاپ شود:

```
insufficient drive size
```

اگر دستور معتبر بود عبارت زیر چاپ شود:

```
paste completed
```

```
cut file #fileName # fileName ...
```

فایل‌ها با اسم داده شده را کات می‌کند (می‌تواند ۱ یا چند فایل باشد). توجه کنید که حتی اگر فقط برای یکی از فایل‌ها ارور زیر رخ دهد، عملیات کات کلاً انجام نمی‌شود.
اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود (حتی اگر یکی از اسم‌ها وجود نداشت)
توجه کنید با اولین بار که به ارور اشاره شده برخورد می‌کنید عملیات آنجا متوقف می‌شود و نیازی نیست برای دیگر اسم‌های داده شده چک کنید که فایلی وجود دارد یا نه

```
invalid name
```

اگر دستور معتبر بود عبارت زیر چاپ شود:

```
files cut completed
```

```
cut folder #folderName #folderName ...
```

فولدرها با اسم داده شده را کپی می‌کند (می‌تواند ۱ یا چند فولدر باشد). توجه کنید که حتی اگر فقط برای یکی از فولدرها ارور زیر رخ دهد، عملیات کات کلاً انجام نمی‌شود.
اگر فولدری با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود (حتی اگر یکی از اسم‌ها وجود نداشت)
توجه کنید با اولین بار که به ارور اشاره شده برخورد می‌کنید عملیات آنجا متوقف می‌شود و نیازی نیست برای دیگر اسم‌های داده شده چک کنید که فولدری وجود دارد یا نه

```
invalid name
```

اگر دستور معتبر بود عبارت زیر چاپ شود:

```
folders cut completed
```

```
print file stats #fileName
```

مشخصات یک فایل را به فرمت زیر چاپ می‌کند.(در صورت معتبر بودن دستور) مثال ۱:

```
mmhg txt
H:\Java Projects\APSpirng98\mmhg
Size: 400MB
Text: hello World!
```

مثال ۲:

```
WallPaper img
E:\Images\WallPaper
Size: 20MB
Resolution: 1000*2000
Extension: jpg
```

اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

```
write text #textFileName
```

باید در فایل متنی با نام داده شده، متنی که داده می‌شود (بلافاصله در دستور بعدی) را بنویسید (متن قبلی موجود در آن پاک می‌شود). توجه کنید که متن داده شده می‌تواند شامل فاصله و کاراکترهای غیر حرف نیز باشد. فرض کنید که حجم فایل تغییر نمی‌کند. اگر فایلی با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر فایلی با نام داده شده در آن مکان وجود داشت اما فایل متنی نبود، باید ارور زیر چاپ شود:

```
this file is not a text file
```

اگر دستور بالا معتبر بود، دستور بعدی آن به طور کامل متن داخل فایل متنی خواهد بود ولی اگر معتبر نباشد دستور بعدی آن نشانگر متن نیست.

```
print frequent folders
```

این دستور در هر جایی از برنامه می‌تواند وارد شود و باید آدرس و تعداد دفعات باز کردن ۵ فولدری را چاپ کنید که بیشتر از دیگر فولدرها برای آن‌ها دستور open را انجام

داده‌اید.(دقت کنید که حداقل یک بار باید وارد این فولدرها شده باشید) اگر تعداد این فولدرها کمتر از ۵ تا بود (مثلاً قبل این دستور فقط دو بار دستور open به صورت معتبر وارد شده بود)، به تعداد موجود چاپ کنید.

توجه کنید که ترتیب چاپ فولدرها به صورت صعودی بر حسب تعداد ورود خواهد بود. اگر تعداد ورود برای دو فولدر یکسان بود به ترتیب حروف الفبا چاپ کنید.(دقت کنید که باید آدرس آن‌ها را از لحاظ حروف الفبا مقایسه کنید)

توجه: واضح است که وقتی فولدری را کات و یا کپی می‌کنید و در جایی دیگر paste می‌کنید تعداد دفعات ورود به آن ۰ می‌شود.

مثال:

```
H:\Java  Projects\APSpirng98 3
E:\Images 2
E:\Images\a 2
E:\videos 2
```

```
print OS information
```

با این دستور اطلاعات سیستم عامل را به شکل مثال زیر چاپ می‌کنید:

```
OS is #name # version
OS is windows 10Pro
```

برنامه در نهایت با دستور زیر به پایان می‌رسد

```
end
```

اگر دستوری وارد شود که در فرمت دستورات بالا نیست ارور زیر را چاپ کنید. توجه کنید که ممکن است دستور نامعتبر هر جایی از برنامه وارد شود.

```
invalid command
```