دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

برنامه‌سازی پیشرفته

نیم‌سال دوم ۹۷-۹۶

مدرس: بهنام حاتمی

تمرین دوم | آشنایی با شئ‌گرایی | مهلت ارسال: ۱۳ فروردین

به موارد زیر توجه کنید:

- پاسخ تمرینات را در سامانه‌ی داوری کوئرا بارگذاری کنید.
- این تمرین ۱۰۰ نمره اجباری دارد و شامل ۱۵ نمره امتیازی برای سؤال آخر است که به همان نسبت در نمره کل تمرین نمره امتیازی دارد(۱۵ درصد).
- حتماً در ارسال فایل برنامه به نام فایل و فرمت ورودی و خروجی‌ها توجه شود. در صورت اشتباه در نحوه‌ی دریافت ورودی و چاپ خروجی، نمره‌ای به شما تعلق نخواهد گرفت.
- هم‌فکری و هم‌کاری در پاسخ به تمرینات اشکالی ندارد؛ ولی پاسخ ارسالی حتماً باید توسط خود شخص نوشته شده‌باشد.
- مبنای درس، اعتماد بر پاسخ ارسالی از سوی شماست؛ بنابراین ارسال پاسخ در سامانه‌ی داوری به این معناست که پاسخ آن تمرین، توسط شما نوشته شده‌است. در صورت تقلب و یا اثبات عدمِ نوشتار پاسخِ حتی یک سوال از تمرین، نمره‌ی آن تمرین به طور کامل برای هر دو طرفِ تقلب‌گیرنده و تقلب‌دهنده برابر صفر قرار داده خواهد شد.
- مهلت ارسال فایل‌ها تا ساعتِ ۲۳:۵۹ سیزدهم فروردین است است. پس از پایان این مهلت، تا ۴ روز به ازای هر روز تاخیر ۱۰ درصد از نمرَه مربوط به تمرین کسر خواهد شد و پس از ۴ روز نمره‌ای به تمرین تعلق نخواهد گرفت.
- دقت داشته باشید که تمیز بودن کد هر سؤال بخشی از نمره تحویل حضوری تمرین است، لذا سعی کنید اصول مربوط به تمیز بودن کد را که در کارگاه کدنویسی تمیز آموزش داده شد را رعایت کنید. تحویل حضوری تمرین هفته بعد از اتمام مهلت ارسال آن خواهد بود.
- معیارهای بررسی تمیز بودن کد شامل این موارد هستند: نام‌گذاری معنادار و با قالب صحیح، قابل قبول بودن اندازه متدها(عدم وجود متدهایی با حجم بسیار زیاد)، عدم وجود کد کپی و پیست‌شده و رعایت فاصله‌گذاری مناسب در خطوط کد و دندانه‌گذاری(Indentation) صحیح.
- توجه داشته باشید که از جایی که این تمرین با محوریت مفاهیم شئ‌گرایی طراحی شده است، بخش قابل توجهی از نمره تمرین به شئ‌گرا بودن کد شما بستگی دارد و این مورد در تحویل حضوری تمرین بررسی خواهد شد، لذا گرفتن نمره کامل در سامانه کوئرا به معنای کامل شدن نمره تمرین نخواهد بود و کد شما باید استانداردهای شئ‌گرایی را نیز رعایت کرده باشد.
- هرگونه سؤال مربوط به تمرین‌ها را با موضوع مناسب در کوئرای درس و در پست همین تمرین مطرح کنید.

سؤال ۱. مرکز خرید (۳۵ نمره)

در این سوال باید یک مرکز خرید بزرگ را شبیه سازی کنید. مرکز خرید به این صورت است که شامل تعدادی مغازه میباشد. هر مغازه تعدادی جنس دارد. و از هر جنس چند نمونه محدود. مشتری ها با ورود به مغازه خرید میکنند و با هر بار خرید، تعداد آن جنس کم میشود. همچنین مغازه دار میتواند جنس خریده و با اینکار تعداد جنس مورد نظر را زیاد کند.

دستورات ورودی

- `add store_1 productA 2 2 productB 3 1 productC 2 1 productD 7 1 productE 6 2`

  این دستور به این معنی است که مغازه ای به نام store_1 در این فروشگاه افتتاح میشود. پس از آمدن اسم مغازه به ترتیب، نام محصولات موجود در مغازه، تعداد آن ها و قیمت آن ها آورده میشود. برای مثال در این مغازه، از محصول productE که قیمت آن ۲ واحد پول است، ۶ نمونه داریم. دقت کنید از سری ورودی های غلط این است اگر پس از نام مغازه، محصولات در دستور وارد نشوند، شما نباید این مغازه را ایجاد کنید. به عبارت دیگر، مغازه ای بدون محصول ساخته نمیشود.

- `buy from store_1 productC 2`

  یک مشتری از مغازه store_1، دو تا از کالای productC خریداری میکند. دقت کنید دیگر این مغازه این جنس را ندارد و اگر دوباره با دستور خرید این جنس رو به رو شود باید پیغام we don't have it(همگی با حروف کوچک) چاپ شود.(اگر محصول درخواستی جز محصولات مغازه نبود و یا به آن تعداد محصول نداشت باید این عبارت چاپ شود). پس از اجرا شدن این دستور، مشتری مبلغ "تعداد * قیمت محصول درخواستی" را میپردازد و به صندوق مغازه این مقدار اضافه میشود. دقت کنید که در این دستور، قیمت محصول به شما داده نشده و باید از قیمتی که فروشگاه روی محصول گذاشته است استفاده کنید.

- `buy for store_1 productF 12 9`

  به این معنی است که این مغازه از کالای productF، که قیمت هر نمونه آن ۹ واحد پول است، ۱۲ نمونه خریداری میکند. پس از اجرای این دستور مبلغ ۱۲*۹ از صندوق مغازه کم میشود. دقت کنید که اگر صندوق مغازه این مقدار پول را برای پرداخت نداشته باشد باید عبارت not enough money چاپ شود. همچنین در صورت اجرا شدن این خط دستور، این محصول به محصولات مغازه اضافه میشود.

- `change price productA of store_1 5`

  این دستور به این معنی است که ازین پس قیمت محصول productA از مغازه store_1 عوض شده و برابر با ۵ واحد پول میگردد.

- `show store_1`

  با وارد شدن این دستور، تمامی اجناس این فروشگاه و تعداد و قیمت آن ها به ترتیب صعودی بر اساس تعداد موجودی چاپ شوند، در صورتی که موجودی دو محصول برابر بود، آن که از لحاظ الفبایی زودتر می‌آید چاپ شود.

```
ProductA 2 5
productB 3 1
productE 6 2
productD 7 1
productF 12 9
```

دقت کنید چون جنس productC تمام شده است، نباید عبارت productC 0 1 چاپ شود.

همچنین هر مغازه ای که افتتاح میشود در صندوقش ۲۰ واحد پول دارد. به گرفتن دستورات تا زمانی ادامه دهید که دستور end در ورودی بیاید.

خطاهای ورودی و پیام‌های خطا

- اسم هیچ مغازه ای با عدد شروع نمیشود. در صورت وارد شدن ورودی به این صورت، نباید این مغازه را اضافه کنید.
- اگر در دستوری با اسم مغازه ای رو به رو شدید که در مرکز خرید وجود ندارد باید عبارت no such store را خروجی بدهید.
- اگر در دستور اضافه کردن مغازه، قیمت محصولی یا تعداد آن وارد نشود پیغام invalid input را چاپ کنید.
- در دستور اضافه کردن مغازه، اگر نام محصولی دو بار آورده شود، خطا بوده و باید از انجام این دستور صرف نظر شود و عبارت invalid input چاپ شود.
- در دستور خرید اجناس برای مغازه، اگر تنها یک عدد وارد شود، خطا بوده و نباید کاری انجام شود و باید عبارت invalid input را چاپ کنید.
- در دستور تغییر قیمت اجناس، در صورت نیامدن قیمت تنها باید عبارت invalid input را خروجی بدهید.

دقت کنید تنها برای موارد ذکر شده خروجی invalid input را چاپ کنید و در صورتی که دستوری یک یا بیش از یکی از مواردی که منجر به خروجی invalid input را در خود داشت، همین خروجی را چاپ کنید.

نکات شئ‌گرایی

که در این سوال، تمامی کلاس ها باید مطابق با نمودار UML داده شده پیاده سازی شوند. در صورتی که نیاز به پیاده‌سازی یک متد یا کلاس کمکی که در UML ذکر نشده دارید، می‌توانید آن را پیاده کنید اما چارچوب کلی کد شما بایستی مبتنی بر UML داده‌شده باشد و باید متدها و متغیرهای آن را پیاده کنید.

نمودار UML کلاس‌ها

«ShoppingCenter»
- stores : ArrayList<Store>
+ addStore(storeName : String, products : HashMap<Product, Integer>) : void + show(storeName : String) : void + sell(storeName : String, productName : String, number : int) : void + buy(storeName : String, productName : String, number : int, price : int) : void + changePrice(storeName : String, productName : String, newPrice : int) : void

«Store»
- name : String - money : int - products : HashMap<Product, Integer>
+ setName(name: String) : void + getName( ) : String + start(storeName : String, products : HashMap<Product, Integer>) : void + getProducts( ) : HashMap<Product, Integer> + changePrice(productName : String, newPrice : int) : void + buy(product : Product, number : int) : void + sell(productName : String, number : int) : void + show( ) : void

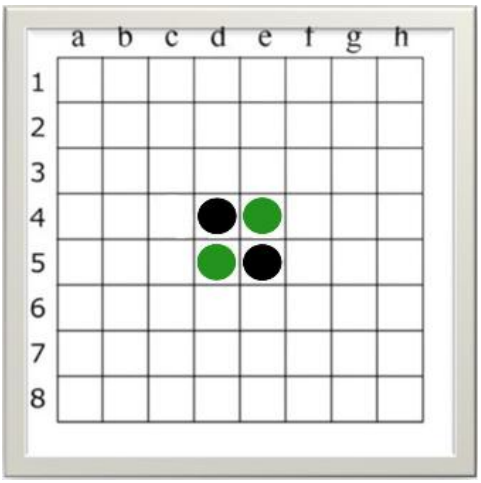
«Product»
- name : String - price : int
+ setName(name: String) : void + getName( ) : String + setPrice(price: int) : void + getPrice( ) : int

نمونه‌ی ورودی و خروجی

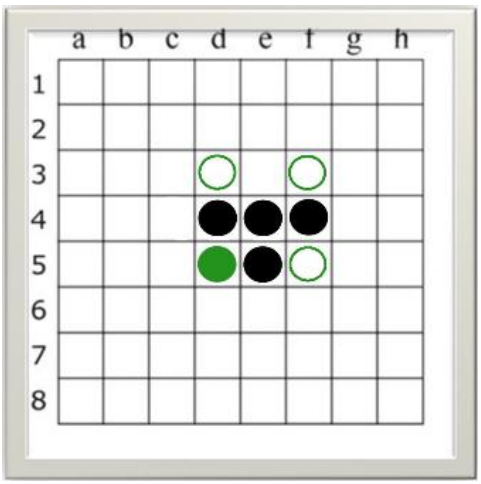
نمونه‌ی ورودی	نمونه‌ی خروجی
add store_1 p1 1 1 p2 1 2 p3 1 4 p4 1 5 add store_2 p1 1 1 p1 1 3 add store_2 add store_2 p1 1 p2 3 4 add store_2 p1 10 5 p2 3 5 p5 8 1 buy from store_1 p1 1 buy from store_1 p2 1 add 3store p6 3 2 buy for 3store p7 1 2 buy for store_2 p3 4 buy for store_1 p1 3 1 change price p3 of store_1 5 show store_1 show store_2	 invalid input invalid input no such store invalid input p3 1 5 p4 1 5 p1 3 1 p2 3 5 p5 8 1 p1 10 5

سؤال ۲. اتللو (۳۵ نمره)

در این قسمت شما باید شبیه ساز یک بازی اوتللو را پیاده سازی کنید. خلاصه قوانین بازی به صورت زیر است:
برای شروع، چهار مهره مطابق شکل در وسط صفحه به صورت ضربدری، قرار می‌گیرند.

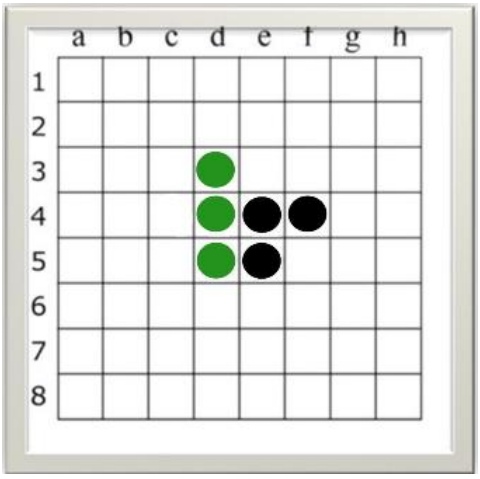


روش بازی
مهره تیره بازی را آغاز می‌کند. هر یک از دو بازیکن به نوبت یک حرکت انجام می‌دهند. مهره را جایی قرار دهید که یک یا چند مهره حریف را محاصره کند. انجام حرکت به معنی گذاشتن یک مهره (از طرف رنگ خود) در صفحه و محصور کردن یک یا چند مهره حریف در یک یا چند راستا است. در نتیجه مهره‌های محاصره شده را برگردانده و به رنگ مهره خود درآورید. این به معنی محاصره و تصاحب است.
برای مثال همان طور که گفته شد سیاه حرکت اول را انجام می‌دهد. مطابق شکل سیاه می‌تواند مهره خود را در یک از خانه‌های E3، F4، C5 یا D6 بگذارد. فرض کنیم سیاه مهره‌اش را در خانه F4 قرار دهد. مهره سبزی که بین دو مهره سیاه قرار گرفته، برگردانده می‌شود و به رنگ سیاه در می‌آید. حالا نوبت سبز است که بازی کند. مطابق شکل سبز می‌تواند مهرهٔ خود را در یکی از خانه‌های E3، C3 و یا C5 قرار دهد.



سبز مهره خود را در خانه D3 قرار می‌دهد. مهره سیاهی که بین دو مهره سبز قرار گرفته برگردانده می‌شود.

بازی به همین ترتیب ادامه می‌یابد و بازیکنان به نوبت با گذاشتن مهره خود در محل‌های مجاز مهره‌های حریف را در راستاهای مختلف محصور کرده و تصاحب می‌کنند. هدف داشتن بیشترین مهره رنگ خود روی صفحه در پایان بازی



است.

پایان بازی

وقتی تمام صفحه پر شود و یا هیچ کدام از دو طرف حرکتی نداشته باشند، بازی به پایان می‌رسد و بازیکنی که تعداد مهره‌های بیشتری روی صفحه بازی داشته باشد برنده می‌باشد.

نکته: ممکن است بازی قبل از پر شدن همه خانهها به پایان برسد. البته در این جا برای راحتی کار، جدول را ۶ در ۶ در نظر گرفتهایم.

برنامه شما باید قابلیت‌های زیر را داشته باشد:

- فضایی برای بازی که بازیکنان بتوانند در آن بازی کنند! برنامه باید قدرت این را داشته باشد که بعد از گذاشتن مهره توسط بازیکن، مهره‌هایی را که باید تغییر رنگ دهند را شناسایی کند و این کار را انجام دهد. علاوه بر این باید زمانی که یک بازیکن نمی‌تواند حرکتی انجام دهد، از نوبت او بپرد. هم چنین قابلیت داشته باشد تا پایان بازی را شناسایی کرده و برنده و بازنده را مشخص کند و بازی را پایان ببخشد.
- باید بتوان یک بازی را ذخیره کرد و بعدا ادامه آن را بازی کرد.
- بخشی باید داشته باشد که در آن لیست بازیکن ها را مشاهده و مرتب‌سازی کرد برنامه باید قابلیت مرتب‌سازی بر اساس نام و تعداد دست برده را داشته باشد. در این بازی هر بازیکن یک پروفایل مخصوص به خود را دارد که حاوی اطلاعاتی مثل تعداد دست‌های بازی کرده و تعداد دست‌های برده است.
- در این بازی در هر دست هر بازیکن میتواند سه مرتبه از گزینه undo استفاده کند. به این صورت که پس از گذاشتن مهره، تا زمانی که حریف مهره خود را نگذاشته میتواند مهره گذاشته شده را بردارد و در جای دیگری بگذارد.

دستورات ورودی

- `new player player_name`

  با وارد کردن این دستور یک بازیکن جدید به نام player_name به سیستم اضافه میشود. در صورتی که نام انتخاب شده قبلا استفاده شده باشد پیغام زیر چاپ می‌شود:

  ```
  name player_name is already used
  ```

- `print players`

  با وارد کردن این دستور بازیکنان و اطلاعات آنان به ترتیب دست برده تا اکنون نمایش داده میشوند. اگر دست برده دو بازیکن برابر بود، کسی که تا کنون دست کم‌تری بازی کرده ابتدا چاپ میشود. در صورت برابر بودن

این دو مورد، افراد به ترتیب لغت نام‌های مرتب میشوند. در هر سطر به ترتیب نام یک بازیکن، تعداد دست‌های بازی‌کرده و تعداد دست‌های برده چاپ میشوند. برای مثال خروجی میتواند به شکل زیر باشد.

```
reza 14 14
ali 15 14
hassan 19 10
```

- `new game player_name1 player_name2 game_name`

با وارد کردن این دستور، ابتدا بازیکن‌ها در بازیکنان و نام بازی انتخابی در بازی‌های ذخیره شده چک میشوند. اگر بازیکنی به این نام وجود نداشت یا اسم بازی قبلا استفاده شده باشد، پیغام زیر را چاپ میکند:

```
invalid name
```

اما اگر نام‌ها درست باشند، وارد فضای بازی میشویم. صفحه بازی یک مربع ۶*۶ است. بازیکنی که در ابتدا نامش نوشته شده باشد شروع کننده بازی و دارای رنگ مشکی است. بعد از شروع بازی، در هر مرحله ابتدا جدول بازی و سپس نام بازیکنی که باید حرکت را انجام دهد چاپ می‌شود. سپس بازیکن، آدرس خانه مورد نظر را وارد میکند. آدرس خانه بدین صورت است که ستون‌ها از چپ به راست از A با حروف انگلیسی و سطرها از بالا به پایین با اعداد از ۱ نام‌گذاری میشوند. در صورتی که بازیکن بتواند در آن خانه مهره بگذارد(بازیکن تنها می‌تواند در خانه‌هایی مهره بگذارد که در صورت گذاشتن مهره در آن خانه در بقیه خانه‌ها تغییری ایجاد شود، یعنی نمی‌توان در خانه‌ای مهره گذاشت که چیزی عوض نشود، برای مثال خانه‌ای که کنارش هیچ مهره‌ای وجود ندارد)، مهره گذاشته شده سپس جدول و نام بازیکنی که در نوبت بعدی مهره میگذارد چاپ میشود. در غیر این صورت و در صورتی که فرمت نام خانه وارد شده به صورت یک حرف بزرگ الفبای انگلیسی و یک عدد نباشد یا چنین خانه‌ای وجود نداشته باشد پیام زیر چاپ می شود:

```
invalid choice
```

نحوه چاپ کردن جدول بدین صورت است که خانه‌های خالی با ـ خانه‌های سیاه با 0 و خانه‌های سفید با 1 نمایش داده میشوند. دقت کنید بین هر دو خانه یک فاصله(space) وجود دارد. در صورتی که دستور undo وارد شود، جدول به حالت قبل بازمیگردد و چاپ می‌شود و دوباره نوبت بازیکن قبلی میشود. از undo زمانی که حریف از آن استفاده کرده نمی‌توان استفاده کرد. یعنی undo توانایی برگشتن به یک مرحله قبل را دارد. در صورت وارد کردن undo زمانی که حریف undo کرده یا حالت قبلی وجود ندارد یا بازیکن از هر سه مرتبه undo استفاده کرده باشد پیام زیر چاپ می‌شود:

```
invalid undo
```

سپس دوباره جدول و نام بازیکنی که باید حرکت را انجام دهد چاپ می شود. با وارد کردن دستور quit از فضای بازی خارج میشویم. اگر بازی هنوز به پایان نرسیده باشد، بازی ذخیره می شود. هر دستور به جز این دستورات در این فضا عمل نمی‌کند و پیام invalid command چاپ میشود.
بازی تا جایی ادامه پیدا میکند که دیگر بازیکنی امکان حرکت نداشته باشد. در این صورت با توجه به تعداد مهره‌های هر بازیکن در زمین برنده شناسایی میشود و بعد از چاپ پیغام زیر از فضای بازی خارج میشویم و بازی از لیست بازی‌های ذخیره شده خارج میشود.

```
player_name won
```

- `load game`

  با وارد کردن این دستور، لیستی از بازی‌ها به ترتیب از قدیم به جدید آورده می‌شود. در این لیست تنها بازی‌های ناتمام وجود دارند. چاپ شدن لیست بازیکنان بدین‌گونه است که در هر خط ابتدا نام بازی سپس نام بازیکن اول و دوم می‌آید. سپس بازیکن نام بازی مورد نظر را وارد میکند. سپس وارد بازی انتخابی میشود. اما اگر نام مورد نظر موجود نباشد، پیام زیر چاپ شده و دوباره بازی‌ها چاپ می‌شوند.

  `invalid name`

  در ضمن در این فضا تنها می‌توان نام بازی یا دستور quit را وارد کرد. هر ورودی دیگر منجر به پیام بالا میشود. برای خارج شدن از این منو، کاربر باید دستور quit را وارد کند.

- `end`

  با این دستور هر جا که باشیم برنامه پایان می‌پذیرد و از آن خارج می‌شویم.

در صورت وارد کردن دستور اشتباه(دستوری که به قالب هیچ‌یک از دستورهای بالا نباشد و شامل هیچ‌یک از خطاهای ذکرشده در هر دستور نیز نباشد) در منوی اول، برنامه باید پیام زیر را چاپ کند.

`invalid command`

برای مثال در صورت گرفتن دستور dsafkjsdaf asdjkad (!) باید در خروجی invalid command چاپ کنید. همین‌طور دقت داشته باشید پس از اتمام بازی به منوی اصلی بازمی‌گردیم.

نکات شئ‌گرایی

برای پیاده‌سازی این سؤال باید از الگوی MVC استفاده کنید، در غیر این صورت هم کارتان بسیار سخت خواهد شد و هم نمره شئ‌گرایی کد این سؤال را نخواهید گرفت.

راهنمایی

برای ساده‌تر شدن کار پردازش ورودی‌ها و دستورهای غلط از رجکس استفاده کنید.

نمونه‌ی ورودی و خروجی

نمونه‌ی ورودی	نمونه‌ی خروجی
new player ali new player hassan sadvlkfb new game ali hassan game1 C5	invalid command - - - - - - - - - - - - - - 1 0 - - - - 0 1 - - - - - - - - - - - - - - ali: - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - hassan:

ادامه‌ی تست‌کیس:

نمونه‌ی ورودی	نمونه‌ی خروجی
D5 undo undo quit print players new game hassan ali game2 quit load game a game1 quit end	- - - - - - - - - - - - - - 1 0 - - - - 1 0 - - - - 1 0 - - - - - - - - ali: - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - hassan: invalid undo - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - hassan: ali 0 0 hassan 0 0 - - - - - - - - - - - - - - 1 0 - - - - 0 1 - - - - - - - - - - - - - - hassan: game1 ali hassan game2 hassan ali invalid name game1 ali hassan game2 hassan ali - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - hassan:

نمونه‌ی ورودی	نمونه‌ی خروجی
new player a new player b load game lsdfjsdf quit lfdjdslfkj new game a b game1 C5 D5 undo undo D2	invalid name invalid command - - - - - - - - - - - - - - 1 0 - - - - 0 1 - - - - - - - - - - - - - - a: - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - b: - - - - - - - - - - - - - - 1 0 - - - - 1 0 - - - - 1 0 - - - - - - - - a: - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - b: invalid undo - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - b: invalid choice - - - - - - - - - - - - - - 1 0 - - - - 1 1 - - - - 1 - - - - - - - - - b:

ادامه‌ی تست‌کیس:

نمونه‌ی ورودی	نمونه‌ی خروجی
D5 quit load game sdfgvfdv game1 E3 quit end	- - - - - - - - - - - - - - 1 0 - - - - 1 0 - - - - 1 0 - - - - - - - - a: game1 a b invalid name game1 a b - - - - - - - - - - - - - - 1 0 - - - - 1 0 - - - - 1 0 - - - - - - - - a: - - - - - - - - - - - - - - 1 1 1 - - - 1 1 - - - - 1 0 - - - - - - - - b:

سؤال ۳. گاوداری (۳۰ نمره + ۱۵ نمره امتیازی)

سال گذشته حسنی در رشته مهندسی کامپیوتر در دانشگاه شریف قبول شد. حالا بعد از سه ترم تحصیل او میخواهد برای این که خودش و توانایی‌های علم کامپیوتر را به پدرش نشان دهد، یک برنامه مدیریت گاوداری برای گاوداری پدرش بنویسد. اما او که در این سه ترم چیزی یادنگرفته تصمیم گرفته این کار را به عنوان تکلیف درس AP به دستیاران آموزشی بدهند تا آنها هم به شما بدهند. حالا شما باید این برنامه را پیاده سازی کنید. در ادامه اطلاعات لازم برای این کار آمده است.

- **گاوداری(Dairy Farm)**

گاوداری از یک انبار شیر، شامل تعدادی مخزن شیر با ظرفیت‌های مشخص، تعدادی باربند، که در هر یک تعدادی گاو قرار می‌گیرند و یک انبار آذوقه با ظرفیتی مشخص تشکیل شده است.

- **باربند(!Barband)**

قسمتی از گاوداری است که گاوها در آن نگهداری میشوند. در هر باربند تعداد مشخصی گاو وجود دارد. در ابتدا باربندی وجود ندارد. کاربر میتواند باربند اضافه کند.

- **گاوها(Cows)**

موجوداتی مهربان هستند که به ما شیر و گوشت میدهند. هر گاو در هر روز مقداری غذا میخورد و شیر میدهد(در ادامه درباره چگونگی این موضوع توضیح داده شده است). هر گاو پس از اضافه شدن به یک باربند یک شماره مخصوص بین تمامی گاوها میگیرد. این شماره به این صورت است که اولین شماره خالی (شروع از یک) به گاو داده می‌شود. هر گاو متاسفانه تنها ۵۰ روز عمر میکند و از فردای اضافه شدن شروع به شیر دادن میکند.

در ابتدا هر گاو ۲۰۰ کیلو وزن دارد و ۵ کیلو غذا در روز نیاز دارد و متناسب با غذای خورده شده شیر تولید میکند. با گذشت هر ۱۰ روز، ۱ یک کیلو به غذای لازم اضافه میشود. با گذشت هر روز مقدار گرسنگی گاو(میزان غذایی که در یک روز نیاز داشته و نتوانسته بخورد) به روز بعد انتقال می‌یابد. هر گاه گرسنگی گاو برابر ۴ برابر غذای لازم در یک روز شود گاو می‌میرد. اگر گاوی دوشیده نشود، فردا به همان اندازه‌ای که امروز شیر داشته به علاوه مقدار شیری که تا فردا تولید میکند، شیر می‌دهد. اما این مقدار از حداکثر توان گاو بیشتر نمی‌شود. حداکثر توان گاو در ابتدا ۲۵ کیلو است و هر ۱۰ روز ۵ کیلو به این مقدار اضافه میشود. هرگاه یک گاو ۳ روز پشت سر هم دوشیده نشود دیگر شیر نمی‌دهد.

خوراک‌دهی به این صورت است که کاربر شماره یک باربند و لیستی از انواع خوراک‌ها و مقدار هر کدام از آنها را وارد میکند. سپس لیست داده شده به آخورِ باربند مورد نظر ریخته میشود.

از جایی که گاوها موجودات مهربانی هستند، به ترتیب گرسنگی شروع به خوردن میکنند (اول گرسنه‌ترها). اگر چند گاو گرسنگی برابر داشته باشند، به ترتیب سن(اول جوان‌ترها) و اگر سن برابر داشته باشند بر اساس شماره(کم به زیاد) شروع به خوردن می‌کنند. هر گاو یک‌واحد غذا می‌خورد و نوبت را بر اساس معیار بالا به نفر بعد از خود می‌دهد، زمانی که همه‌ی گاوها یک واحد غذا خوردند دوباره این روند تکرار می‌شود، تا جایی که همه‌ی گاوها سیر شوند یا غذا تمام شود.

اگر مقدار غذای داده‌شده بیش از نیاز باشد، مقدار اضافی در آخور باقی میماند و فردا مصرف میشود. اگر گاوی با شکم گرسنه بخوابد، به اندازه غذای مورد نیاز از وزن او کم میشود. اگر وزن گاوی کم تر از ۱۰۰ کیلو شود، گاو می میرد.

- **خوراک‌ها(Feed)**

  - **جو(Barley)**: به ازای خوردن هر کیلو جو ۴ لیتر شیر تولید میشود و ۴ کیلو به وزن گاو اضافه میشود. درجه علاقه گاوها به جو ۸۰ است.

- **یونجه(Alfalfa)**: به ازای خوردن هر کیلو یونجه ۳ لیتر شیر تولید میشود و ۳ کیلو به وزن گاو اضافه میشود. درجه علاقه گاوها به یونجه برابر ۶۰ میباشد.
- **کاه(Straw)**: به ازای خوردن هر کیلو کاه ۲ لیتر شیر تولید میشود و وزنی به گاو اضافه نمی‌شود. درجه علاقه گاوها به کاه ۲۰ است.

دقت داشته باشید اضافه شدن وزن گاو تنها به صورت بالا صورت می‌پذیرد و شیری که گاو در خود دارد و غذایی که می‌خورد قسمتی از وزن گاو محسوب نمی‌شود.

این که بین چند غذا گاوها کدام یک را اول میخورند توسط درجه علاقه مشخص میشود. بعنوان مثال وقتی چند نوع خوراک در آخور باشد، گاوها شروع به خوردن آن غذایی می‌کنند که بیشتر دوست دارند. تا سیر شوند یا آن غذا تمام شود(در صورت وجود نوع دیگر به سراغ غذای بعدی میروند).

- **انبار خوراک‌ها(Storage)**

گاوداری انباری دارد که خوراک‌ها در آن انبار می‌شوند. این انبار ظرفیتی دارد که می‌توان آن را افزایش داد. دقت کنید ظرفیت اولیه انبار صفر است.

- **مخازن شیر(Tanks)**

این قسمت از چندین مخزن شیر با ظرفیت‌های مشخص تشکیل شده است. شیرها بعد از دوشیده شدن سه روز ماندگاری دارند و در روز چهارم فاسد می‌شوند. درصورتی که شیر فاسد شود تمام شیر درون مخزن را آلوده می‌کند. بعنوان مثال اگر در روز اول مقداری شیر درون مخزن ریخته شود و در روزهای بعد(دوم و سوم) به این مقدار اضافه شود تمام شیرها در ابتدای روز چهارم فاسد میشوند.

دستورات ورودی

در ابتدا برنامه پیام زیر را چاپ میکند:

```
set date
```

سپس کاربر تاریخ را به فرمت yyyy/mm/dd وارد میکند. بعد از تنظیم تاریخ برنامه آماده پذیرش ورودی میشود. در صورت ورود تاریخ یا ورودی نامعتبر(منظور از ورودی نامعتبر ورودی است که به فرم بالا نباشد و منظور از تاریخ نامعتبر تاریخی است که روزش از ۳۰ بیشتر یا ماهش از ۱۲ بیشتر باشد، همین‌طور عدد سال نیز نباید با رقم صفر شروع شود)، برنامه پیام زیر را چاپ میکند.

```
invalid date
```

همین‌طور دقت کنید برای سادگی بیشتر همه‌ی ماه‌ها ۳۰ روزه در نظر گرفته شده‌اند.

- `add barband n`

این دستور یک باربند با ظرفیت n به گاوداری اضافه می‌کند. باربند اضافه شده اولین شماره خالی با شمارش از یک را به خود اختصاص می‌دهد.

- `status barband x`

اطلاعات باربند x شامل تعداد گاوها، ظرفیت و میزان باقی‌مانده از هر نوع غذا را نمایش میدهد. غذاها به ترتیب علاقه گاوها به غذاها چاپ می‌شوند، به ترتیب نزولی. دقت کنید برای مثال خروجی به صورت زیر است:

```
barband 3
number of cows: 5
capacity: 50
the remaining food:
```

```
barley 10
alfalfa 3
cows:
1
3
4
```

در صورتی که باربندی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid barband
```

- `status cow n`

اطلاعات گاو به شماره x شامل سن، گرسنگی، وزن، میزان شیری که دارد و میزان شیری که تا به حال تولید کرده را نشان میدهد برای مثال:

```
cow 120
age: 39
hunger: 4
weight: 550
milk: 35
milk produced: 150
```

در صورتی که گاوی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid cow
```

- `add cow x`

یک گاو به باربند شماره x اضافه میکند. بر اساس این که باربند جا داشته باشد یا نداشته باشد یکی از پیغام‌های زیر چاپ میشود. برای مثال :

```
cow added. cow num: 21
there is not enough space in barband 4
```

- `add tank n`

مخزن شیری با ظرفیت n لیتر به مخازن شیر اضافه می‌کند.

- `status farm`

اطلاعات گاوداری شامل تعداد گاوها، باربندها، مخزنهای شیر، حداکثر ظرفیت تولید شیر در روز(برابر با جمع مقدار شیر گاوها در هر روز در بهترین حالت یعنی تمام گاوها به اندازه حداکثر ظرفیتشان شیر داشته باشند) و ظرفیت انبار را نشان می‌دهد. برای مثال:

```
dairy farm
number of barbands: 6
number of cows: 340
storage capacity: 11000
milk tank num: 6
max milk production: 400
```

- `feed barband x`

بعد از زدن این دستور یک لیست از خوراک‌ها و مقدار آنان می‌آید و در آخر end_feed میآید. سپس لیست مورد نظر به باربند ریخته میشود. دقت کنید بلافاصله پس از ریخته شدن غذا به باربند گاوها بر اساس نحوه ذکر شده شروع به خوردن غذا می‌کنند.

```
feed berband 3
alfalfa 30
barley 20
straw 50
end_feed
```

اگر در خطی ورودی نامعتبر(همانند خوراکی که موجود نیست) بیاید، برنامه آن خط را در نظر نمیگیرد. هم‌چنین در صورتی که باربندی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid barband
```

- `status storage`

این دستور اطلاعات انبار شامل ظرفیت آن و میزان خوراکی موجود از هر نوع را در قالب زیر نشان خواهد داد. دقت کنید غذاها به ترتیب علاقه‌ی گاوها به آن‌ها چاپ می‌شوند، به صورت نزولی.

```
storage
capacity: 1100
inventory:
barley 2
alfalfa 400
```

- `status tanks`

این دستور اطلاعات مخزن‌ها را در قالب زیر نشان خواهد داد:

```
tank 1
capacity: 300
empty space: 20
expiration date: 1377/04/19
tank 2
…
```

در صورتی که مخزنی شیر نداشت تاریخ انقضایش را بی‌نهایت در نظر بگیرید و در دستور بالا جلوی expiration date چیزی چاپ نکنید.

- `milk cow n m`

دوشیدن گاو شماره n و ریختن شیر در مخرن شماره m. خروجی این دستور بنا بر این که گاو شیر داشته باشد یا نه و گنجایش مخرن یکی از پیامهای زیر است.

```
cow milked successfully
there is not enough space
cow has no milk
```

در صورتی که گاو/مخزنی با چنین شماره‌ای وجود نداشته باشد پیام زیر به تناسب در خروجی چاپ می شود:

```
invalid cow/tank
```

- `add storage food_name amount`

مقدار amount کیلوگرم food_name را به انبار اضافه میکند. در صورتی که غذایی با این اسم تعریف نشده باشد پیام زیر در خروجی چاپ می‌شود:

```
invalid food name
```

همین‌طور در صورتی که انبار ظرفیت کافی را نداشته باشد پیام زیر در خروجی چاپ می‌شود:

```
there is not enough space
```

- `sell milk m x`

مقدار m لیتر شیر از مخزن x فروخته میشود. بعد از این دستور یکی از دو پیام زیر چاپ میشود(در صورتی که شیر مخزن فاسد شده باشد نیز همین پیغام خطا چاپ می‌شود).

```
there is not enough milk
milk sold successfully
```

در صورتی که مخزنی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid tank
```

- `empty tank x`

تانک شماره x را خالی میکند. در صورتی که مخزنی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid tank
```

- `butcher cow n`

گاو با شماره n به کشتارگاه فرستاده میشود. در صورت موفق بودن پیام زیر نمایش داده میشود:

```
cow butchered successfully
```

در صورتی که گاوی با چنین شماره‌ای وجود نداشته باشد پیام زیر در خروجی چاپ می شود:

```
invalid cow
```

- `move cow n m`

گاو شماره n را به باربند m انتقال میدهد. در صورت موفق بودن پیام زیر نمایش داده میشود.

```
cow moved successfully
```

در صورتی که گاو/باربندی با چنین شماره‌ای وجود نداشته باشد پیام زیر به تناسب در خروجی چاپ می شود:

```
invalid cow/barband
```

و همین‌طور در صورتی که باربند به اندازه کافی جا نداشته باشد خروجی زیر چاپ خواهد شد:

```
there is not enough space in barband x
```

- `show ranks`

  گاوهای زنده را به ترتیب شیر دوشیده شده تا کنون لیست کرده و نشان میدهد. در صورت برابر بودن مقدار شیر دوشیده شده، بر اساس شماره از کم به زیاد لیست میشوند. در هر خط ابتدا شماره گاو و سپس مقدار شیر دوشیده شده تا کنون چاپ میشود.

- `increase storage capacity n`

  ظرفیت انبار را n واحد افزایش می دهد.

- `add new food food_name n`
  `a`
  `b`
  `c`

  این دستور به اندازه n کیلو از یک غذای جدید را به انبار اضافه میکند. بعد از زدن این دستور، غذای مورد نظر به اندازه n به انبار اضافه میشود. به ازای هر واحد غذا، a واحد شیر تولید و b واحد به وزن گاو اضافه میشود. این غذا نیز دارای درجه علاقمندی c است.

- `day passed`

  این دستور در انتهای یک روز وارد می‌شود. با وارد کردن این دستور وارد روز جدید می‌شویم. تاریخ یک روز به جلو میرود. گاوها با توجه به غذای مصرف شده در روز قبل شیر می دهند، به اندازه غذای لازم در یک روز به گرسنگی‌شان اضافه میشود، غذاهایی که در آخور از روز قبل مانده توسط گاوها خورده می‌شود و شیرهایی که تاریخ مصرفشان روز قبل بوده فاسد می‌شوند. در کل تمام تغییراتی که باید با تغییر روز انجام شود پس از این دستور اعمال می شوند.
  یعنی ابتدای هر روز(در واقع انتهای روز قبل) و با آمدن دستور day، passed ابتدا گرسنگی گاوها افزایش پیدا می‌کند، سپس باقی‌مانده غذاها از روز قبل خورده می‌شود و سپس گاوهایی که گرسنگی‌شان بیش از حد شده می‌میرند. دقت کنید گاوها روزی که به گاوداری اضافه می‌شوند گرسته نیستند.

- `end`

  با خواندن این دستور برنامه پایان می‌یابد.

همچنین در صورتی که دستوری در قالب دستورهای بالا نبود(برای مثال در حالتی که نام دستور اشتباه بود یا یکی از پارامترهای دستور داده نشده بود) پیام زیر را در خروجی چاپ کنید(**راهنمایی:** باز هم رجکس!):

```
invalid command
```

نکات شئ‌گرایی + نمره امتیازی

پیاده‌سازی این سؤال به صورت اصولی و تمیز با الگوی MVC، ۱۵ نمره امتیازی ذکر شده در تمرین را به همراه خواهد داشت.

نمونه‌ی ورودی و خروجی

نمونه‌ی ورودی	نمونه‌ی خروجی
11/11/11 1111/11/11 add barband 10 add barband 3 add barband 1 add cow 1 add cow 1 add cow 1 add cow 2 add cow 2 add cow 2 butcher cow 2 add cow 3 move cow 1 2 move cow 4 1 status cow 3 status cow 10 increase storage capacity increase storage capacity 3 add storage alfalfa 10 increase storage capacity 50000 add storage alfalfa 1000 add storage barley 1000 add storage strae 1000 add storage strae 1000 add storage straw 1000 add new food goodFood 1000 10 5 90 add new food badFood 1000 0 0 10 add tank 20 add tank 1000 status tanks day passed status barband 1 status barband 2	set date invalid date cow added. cow num: 1 cow added. cow num: 2 cow added. cow num: 3 cow added. cow num: 4 cow added. cow num: 5 cow added. cow num: 6 cow butchered successfully cow added. cow num: 2 there is not enough space in barband 2 cow moved successfully cow 3 age: 0 hunger: 0 weight: 200 milk: 0 milk produced: 0 invalid cow invalid command there is not enough space invalid food name invalid food name tank 1 capacity: 20 empty space: 20 expiration date: tank 2 capacity: 1000 empty space: 1000 expiration date: barband 1 number of cows: 3 capacity: 10 the remaining food: cows: 1 3 4 barband 2 number of cows: 2 capacity: 3 the remaining food: cows: 5 6

ادامه‌ی تست‌کیس:

نمونه‌ی ورودی	نمونه‌ی خروجی
status barband 3 feed barband 1 alfalfa 3 barley 3 goodFood 3 badFood 3 straw 10000 end_feed status cow 1 day passed status cow 1 milk cow 1 1 milk cow 2 1 milk cow 3 1 status tanks sell milk 1 1 day passed sell milk 1 1 day passed sell milk 1 1 day passed sell milk 1 1 day passed sell milk 1 1 status tanks status cow 3 end	barband 3 number of cows: 1 capacity: 1 the remaining food: cows: 2 cow 1 age: 1 hunger: 1 weight: 200 milk: 0 milk produced: 0 cow 1 age: 2 hunger: 6 weight: 211 milk: 17 milk produced: 0 cow milked successfully cow has no milk there is not enough space tank 1 capacity: 20 empty space: 3 expiration date: 1111/11/16 tank 2 capacity: 1000 empty space: 1000 expiration date: milk sold successfully milk sold successfully milk sold successfully milk sold successfully there is not enough milk tank 1 capacity: 20 empty space: 7 expiration date: 1111/11/16 tank 2 capacity: 1000 empty space: 1000 expiration date: invalid cow

نمونه‌ی ورودی	نمونه‌ی خروجی
1111/11/11 add barband 10 add cow 1 add barband 2 add cow 2 day passed day passed day passed status barband 1 day passed status barband 1 add cow 1 add cow 1 add cow 2 increase storage capacity 10000 add barley 1000 add storage barley 1000 feed barband 1 barley 10 end_feed status barband 1 day passed status barband 1 feed barband 2 barley 100 end_feed day passed status barband 2 day passed day passed day passed day passed status cow 3	set date cow added. cow num: 1 cow added. cow num: 2 barband 1 number of cows: 1 capacity: 10 the remaining food: cows: 1 barband 1 number of cows: 0 capacity: 10 the remaining food: cows: cow added. cow num: 1 cow added. cow num: 2 cow added. cow num: 3 invalid command barband 1 number of cows: 2 capacity: 10 the remaining food: barley 10 cows: 1 2 barband 1 number of cows: 2 capacity: 10 the remaining food: cows: 1 2 barband 2 number of cows: 1 capacity: 2 the remaining food: barley 90 cows: 3 cow 3 age: 6 hunger: 0 weight: 300 milk: 0 milk produced: 0

ادامه‌ی تست‌کیس:

نمونه‌ی ورودی	نمونه‌ی خروجی
status barband 2 day passed day passed day passed status cow 3 status barband 2 day passed status barband 2 status cow 3 day passed status barband 2 status cow 3 end	barband 2 number of cows: 1 capacity: 2 the remaining food: barley 70 cows: 3 cow 3 age: 9 hunger: 0 weight: 360 milk: 0 milk produced: 0 barband 2 number of cows: 1 capacity: 2 the remaining food: barley 55 cows: 3 barband 2 number of cows: 1 capacity: 2 the remaining food: barley 50 cows: 3 cow 3 age: 10 hunger: 0 weight: 380 milk: 0 milk produced: 0 barband 2 number of cows: 1 capacity: 2 the remaining food: barley 44 cows: 3 cow 3 age: 11 hunger: 0 weight: 400 milk: 0 milk produced: 0