به نام خدا

درس مبانی برنامه‌سازی

تمرین ۳

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۰۱ـ۰۰

استاد:

علیرضا آقامحمدی

مهلت ارسال:

۱۳اردیبهشت ـ ساعت ۲۳:۵۹:۵۹

مسئول تمرین:

ایمان محمدی

مسئولان تحویل تمرین ۳:

دانیال جهانبانی

طراحان تمرین تمرین ۳:

بهزاد نبوی

مهراد میلانلو

پرنیان رضوی پور

محمد زارع بیدکی

فهرست

نکات قابل توجه ۲

سوالات ۳

سوال ۱. دانیال جابز ۳

سوال ۲. راه و شهرسازی ۸

سوال ۳. انباردار ۱۷

نکات قابل توجه

- پیش از شروع تمرین، آیین‌نامه مربوط به انجام تمرین‌های درس را با دقت مطالعه کنید.
- برای پاسخ به سوالات این تمرین تنها مجاز به استفاده از شرط، حلقه، عملگرهای بیتی و توابع ورودی و خروجی هستید. پاسخ‌هایی که خارج از این چهارچوب باشند نمره‌ای دریافت نخواهند کرد.
- داوری سوالات به صورت خودکار خواهد بود. پس آنکه فایل پاسخ را ارسال کنید، سامانه به صورت خودکار کد شما را بررسی کرده و در صورتی که پاسخ درست باشد، نمره‌ای به شما خواهد داد. توجه داشته باشید که این نمره نهایی نیست و در صورتی که شرط‌های ذکر شده در صورت سوال را نقض کرده باشید، نمره‌ی سوال را دریافت نخواهید کرد.
- سوالات و ابهامات خود در رابطه با تمرین را در زیر پست مربوطه در کوئرا بپرسید.

سوالات

سوال ۱. دانیال جابز

شرکت اپل برای نسل بعدی آیفون، دانیال را استخدام کرده تا یک دفترچه تلفن برای موبایل طراحی کند. این دفترچه تلفن باید دستورات مختصری برای ذخیره‌ی مخاطب‌ها و ارتباط با آن‌ها داشته باشد. دانیال از شما برای پیاده‌سازی دستورات ساده‌ی این برنامه کمک می‌خواهد. با این دستور، یک مخاطب ایجاد و ذخیره می‌شود. این دستور هیچ خروجی‌ای ندارد.

```
add {name} {phone_number}
```

با استفاده از این دستور می‌توانیم برای مخاطبی که قبلا add شده یک نام خانوادگی set کنیم.

```
set {name} {family_name}
```

ین دستور خروجی‌ای ندارد اما اگر مخاطبی با name داده شده وجود نداشت باید ارور زیر دریافت شود:

```
Invalid Contact Information!
```

تضمین داده می‌شود که در نام، نام خانوادگی یا شماره تلفن مخاطبین فاصله وجود ندارد. همچنین تمام مخاطبان ورودی دارای اسم، فامیل و شماره همراه منحصر به فرد خواهند بود. با دستور call، می‌توانیم از طریق نام یا نام خانوادگی یا شماره تلفن یک نفر که قبلا ذخیره شده با اون تماس برقرار کنیم.

```
call {name | family_name | phone_number}
```

اگر مخاطبی با اطلاعات داده‌شده وجود نداشت باید ارور زیر دریافت شود:

```
Invalid Contact Information!
```

همچنین بعد از اجرای دستور ،call در صورتی که قبل از اتمام تماس (با دستورhang_up) هرگونه دستوری وارد شود باید ارور زیر دریافت شود:

```
You Must End Your Call First!
```

با اجرای این دستور، تماس خاتمه یافته و time که به عنوان ورودی (به ثانیه) داده می‌شود، به مجموع زمانی که با این مخاطب تماس داشتیم اضافه می‌شود. تضمین می‌شود که مقدار time همواره یک integer خواهد بود.

```
hang_up {time}
```

همچنین در صورتی که قبل از ورود این دستور تماسی برقرار نبوده باشد باید ارور زیر دریافت شود:

```
You Are Not Talking To Anyone!
```

با این دستور از طریق نام یا نام خانوادگی یا شماره تلفن یک نفر او را از دفترچه تلفن حذف می‌کنیم.

```
delete {name | family_name | phone_number}
```

این دستور خروجی‌ای ندارد ولی در صورتی که مخاطبی با اطلاعات داده‌شده وجود نداشته باشد باید ارور زیر دریافت شود:

```
Invalid Contact Information!
```

با این دستور از طریق نام یا نام خانوادگی یا شماره تلفن یک نفر،مجموع زمانی که تا به

حال با او صحبت شده نشان داده می شود.

```
display_time {name | family_name | phone_number}
```

فرمت خروجی باید به شکل زیر باشد:

```
{hours}:{minutes}:{seconds}
```

به عنوان مثال ۲:۰:۱۲ یعنی دو ساعت و دوازده ثانیه (یا در واقع ۳۶۱۲ ثانیه). اگر مخاطبی با اطلاعات وارد شده وجود نداشت باید ارور زیر دریافت شود:

```
Invalid Contact Information!
```

با اجرای این دستور، تمام مخاطبینی که تا این لحظه add شده‌اند، ابتدا بر اساس نامشان و به ترتیب حروف الفبا مرتب می‌شوند.

```
display
```

خروجی با دو فرمت زیر نمایش داده می‌شوند. مخاطبی که نام خانوادگی ندارد:

```
name: danyal || phone_number: 09123456789
```

مخاطبی که نام خانوادگی دارد:

```
name: danyal || family_name: jahanbani || phone_number: 09123456789
```

```
end
```

با ورود این دستور برنامه خاتمه یافته و متوقف می‌شود.

لازم به ذکر است که اگر دستور اشتباهی وارد شود باید خروجی زیر دریافت شود:

```
invalid command!
```

ورودی نمونه ۱

```
add mmd 09123456789
add dani 09023456789
set dani jahan
hang_up 77
call jahan
hang_up 77
display_time 09023456789
display
end
```

خروجی نمونه ۱

```
You Are Not Talking To Anyone!
0:1:17
name: dani || family name: jahan || phone number: 09023456789
name: mmd || phone number: 09123456789
```

ورودی نمونه ۲

```
add farhad 7777777
call farhad
set farhad majidi
```

```
hang_up 7777
delete majidi
call farhad
end
hang_up 4444
display_time 7777777
emd
end
```

خروجی نمونه ۲

```
You Must End Your Call First!
Invalid Contact Information!
You Must End Your Call First!
3:23:41
invalid command!
```

سوال ۲. راه و شهرسازی

در این برنامه می خواهیم به وزارت راه و شهرسازی کمک کنیم تا شهرها و جاده‌ها را مدیریت کند.

** شهر

هر شهر، یک شناسه یکتا و اسم دارد.

** جاده

هر جاده، دارای یک شناسه یکتا، اسم، شناسه شهر مبدا و شناسه شهر مقصد می‌باشد.

منوی اصلی

در ابتدای شروع برنامه کاربر در این منو قرار دارد و پیام زیر نمایش داده می‌شود:

```
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
```

کاربر از این گزینه‌ها، یکی را انتخاب می‌کند. در صورت انتخاب '۱'، پیام زیر نمایش داده می‌شود:

```
Select a number from shown menu and enter. For example 1 is for help.
```

و دوباره به منوی اصلی بر می‌گردد. هر کدام از گزینه‌های '۲'، '۳' و '۴' که انتخاب شوند، به‌ترتیب وارد منوهای $Delete$ و $Show$ و Add می شود. در صورت انتخاب '۵' هم، اجرای برنامه پایان می‌یابد. اگر ورودی کاربر چیزی غیر از اعداد '۱' تا '۵' بود

خطای زیر نمایش داده می‌شود:

```
Invalid input. Please enter 1 for more info.
```

و برنامه دوباره به منوی اصلی برمی‌گردد.

Add Menu

پس از ورود به منوی Add، باید پیام زیر نمایش داده شود:

```
Select model:
  1. Add another City
  2. Main Menu
```

اگر کاراکتری غیر از ‘۱’ و ‘۲’ وارد شوند خطای زیر نمایش داده می‌شود:

```
Invalid input
```

و دوباره به منوی Add برمی‌گردد. درصورت انتخاب ‘۱’ باید پیام‌های زیر نمایش داده شود:

```
id:
```

کاربر باید یک عدد ورودی بدهد. تضمین داده می‌شود که کاربر یک عدد ورودی می‌دهد. اگر همچین شناسه‌ای از قبل وجود داشته باشد، شهر جدید باید جایگزین شهر قبلی با این شناسه شود.

سپس پیام زیر نمایش داده می‌شود:

```
name:
```

کاربر باید اسم شهر را وارد کند. سپس در صورت موفقیت‌آمیز بودن ساخت شهر پیام زیر نمایش داده می‌شود:

```
City with id=[id] added!
Select your next action
  1. Add another City
  2. Main Menu
```

اگر '۱' وارد شود شهر دیگری اضافه می‌شود و اگر '۲' وارد شود به منوی اصلی برمی‌گردیم.

اگر بخواهیم جاده‌ی جدیدی اضافه کنیم در منوی Add، '۲' را وارد می‌کنیم. بعد از آن باید پیام زیر نمایش داده شود.

```
id:
```

کاربر باید یک عدد ورودی بدهد. تضمین داده می‌شود که کاربر یک عدد ورودی می‌دهد. اگر همچین شناسه‌ای از قبل وجود داشته باشد، جاده جدید باید جایگزین جاده قبلی با این شناسه شود.

سپس پیام زیر نمایش داده می‌شود:

```
name:
```

کاربر باید اسم جاده را وارد کند. بعد شناسه شهر مبدا خواسته می‌شود:

```
starting:
```

اگر شهری با چنین شناسه وجود نداشته باشد، پیام زیر نمایش داده می‌شود:

```
There is not any city with this id. Please enter valid id.
Select your next action
  1. Add another City
  2. Main Menu
```

بین این دو گزینه یکی انتخاب می‌شود و تضمین می‌شود کاراکتری جز '۱' و '۲' داده نمی‌شود. اگر هم شناسه درستی داده شود، شناسه شهر مقصد خواسته شده و پیام زیر نمایش داده می‌شود:

```
destination:
```

اگر شهری با چنین شناسه وجود نداشته باشد، پیام زیر نمایش داده می‌شود:

```
There is not any city with this id. Please enter valid id.
Select your next action
  1. Add another City
  2. Main Menu
```

Delete Menu

در این منو ابتدا با پیام زیر روبه‌رو می‌شویم:

```
Select model:
  1. City
  2. Road
```

اگر کاراکتری غیر از '۱' و '۲' وارد شوند خطای زیر نمایش داده می‌شود:

```
Invalid input
```

و دوباره به منوی Delete برمی‌گردد. درصورت انتخاب '۱' باید پیام‌های زیر نمایش داده شود:

```
id:
```

کاربر باید یک عددی ورودی بدهد. در صورتی که حذف موفقیت‌آمیز باشد پیام زیر چاپ می‌شود:

```
City:[id] deleted!
```

اگر هم شهری با چنین شناسه‌ای وجود نداشته باشد، خطای زیر نمایش داده می‌شود:

```
City with id [id] not found!
```

در هر دو حالت هم، برنامه به صورت خودکار به منوی اصلی برمی‌گردد.
اگر کاربر بخواهد جاده‌ای حذف کند، در ابتدای منو عدد '۲' را وارد می‌کند و پیام زیر نمایش داده می‌شود:

```
id:
```

کاربر باید یک عددی ورودی بدهد. در صورتی که حذف موفقیت‌آمیز باشد پیام زیر چاپ می‌شود:

```
Road:[id] deleted!
```

اگر هم جاده‌ای با چنین شناسه‌ای وجود نداشته باشد، خطای زیر نمایش داده می‌شود:

```
Road with id [id] not found!
```

در هر دو حالت هم، برنامه به صورت خودکار به منوی اصلی برمی‌گردد.

Show Menu

در این منو هم ابتدا با پیام زیر روبه‌رو می‌شویم:

```
Select model:
  1. City
  2. Road
```

اگر کاراکتری غیر از‘ ۱‘ و ‘۲‘ وارد شوند خطای زیر نمایش داده می‌شود:

```
Invalid input
```

و دوباره به منوی Show برمی‌گردد.
اگر ‘۱‘ انتخاب شود، در هر خط اطلاعات هر شهری به صورت زیر نمایش داده می‌شود:

```
[id] [name]
```

در صورتی هم که ‘۲‘ انتخاب شود، اطلاعات هر جاده‌ای به صورت زیر نمایش داده می‌شود:

```
[id] [name] [starting] [destination]
```

پس از نمایش داده‌ها، برنامه به صورت خودکار به منوی اصلی برمی‌گردد.

ورودی نمونه ۱

```
2
1
0
teh
1
```

```
1
mash
2
2
2
0
jad
5
1
12
jadde
0
1
2
4
2
4
1
5
```

خروجی نمونه ۱

```
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
Select model:
1. City
2. Road
id:
name:
City with id=0 added!
Select your next action
1. Add another City
2. Main Menu
```

```
id:
name:
City with id=1 added!
Select your next action
1. Add another City
2. Main Menu
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
Select model:
1. City
2. Road
id:
name:
starting:
There is not any city with this id. Please enter valid id.
Select your next action
1. Add Road
2. Main Menu
id:
name:
starting:
destination:
Road with id=12 added!
Select your next action
1. Add another Road
2. Main Menu
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
Select model:
1. City
2. Road
```

```
12 jadde teh mash
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
Select model:
1. City
2. Road
0 teh
1 mash
Main Menu - Select an action:
1. Help
2. Add
3. Delete
4. Show
5. Exit
```

سوال ۳. انباردار

انبارداری یکی از مشاغل پیچیده در این روزگار است! حال شما در این سوال، یک انبار و سیستم انبارداری ساده را بیاده‌سازی می‌کنید. انبار محلی برای ذخیره کالاها با شرایط خاص خودشان است(همچون دمای مورد نیاز، اندازه و ...) و انبار دار موظف است کالاهای ورودی را تا حد ممکن(در صورت کمبود جا یا ... نمیتوان همه کالا ها را اسکان داد!) اسکان دهد.

1-Package:

پکیج ها بسته هایی هستند که شامل تعدادی از کالای ما هستند. فرض کنید انباری دارید که یک بخش آن مختص گوشی های موبایل است. هر گوشی موبایل حجم مشخصی دارد. حال ما می‌توانیم با هر تعداد از این گوشی ها یک پکیج بسازیم. نکته: هر پکیج دارای فقط یک نوع کالای به خصوص است. ویژگی های هر پکیج در زیر آورده شده است:

a.name:

نام کالاهای درون پکیج می‌باشد. هر نام برای هر کالا یکتا است و تضمین می‌شود که دو کالا با مشخصات متفاوت(حجم، دما و قیمت)، نام یکسان ندارند.

b.price:

قیمت کالا که یک عدد صحیح و همواره بزرگ‌تر از ۰ است.

c.temperature:

دمایی که کالا در آن می‌تواند سالم بماند که عددی صحیح است.

d.amount:

تعداد کالا در پکیج مدنظر را بیان می‌کند. عددی همواره صحیح وغیر منفی می‌باشد. اگر این عدد صفر باشد به معنای خالی بودن پکیج است.

e.volume:

حجمی که هر واحد کالا اشغال می‌کند را با یک عدد صحیح و بزرگ‌تر از ۰ بیان می‌کند.

نکته: برای مثال اگر بخواهیم بفهمیم که هر پکیج چه حجمی را اشغال میکند، می‌توانیم از رابطه زیر حساب کنیم:

amount * volume

هر انبار دارای یک محل ذخیره‌سازی سه بعدی یا همان storage است. هر محل ذخیره‌سازی به اتاق‌هایی با ظرفیت های گاه متفاوت تقسیم می‌شود.

هر اتاق یا unit دارای سیستم کنترل و ثابت نگه‌داشتن محیط است و می‌توان دمای هر اتاق را ثابت در دمایی خاص قرار داد. بس شما می‌توانید در اتاق بغل گوشی های موبایل(که نیاز به دمای محیط دارند)، سیب هایی را در دمای ۰ درجه‌سانتیگراد نگه‌داری

کنید. زیرا هر اتاق عایق است و تبادل دما با اتاق های بغل ندارد.
هر اتاق می‌تواند حجمی از کالاها را در خود جای دهد.

2-WareHouse:

انبارشامل یک محل ذخیره سازی (Storage) است که در ادامه به آن می پردازیم.
نکته : ما تنها یک محل ذخیره سازی برای انبارمان می توانیم داشته باشیم !

3-Storage:

محل ذخیره‌سازی یا storage از بالا یک جدول مستطیلی بوده که دارای n * m خانه است. هر خانه را اتاق یا unit می نامیم . (در نتیجه هر محل ذخیره سازی n * m اتاق (Unit) دارد.) در ابتدا همه ی اتاق ها دما و گنجایش پیشفرض دارند .

4-Unit:

اتاق ها دارای وِیژگی دما، گنجایش، موقعیت مکانی هستند و پکیج ها در این اتاق ها قرار می گیرند. در ابتدای کار، همه اتاق ها دارای حجم یکسان و دمای یکسان هستند ولی این پارامتر ها قابل تغییر می‌باشند.

فوت‌وفن انبارداری: یکی از معروف‌ترین الگوریتم های انبارداری، "جا خالی نده!" می‌باشد.

در این روش انباردار پروسه ای برای وارد و خارج کردن کالا انجام می‌دهد که در بخش دستورات توضیح داده خواهد شد.

دستورات : مسئولیت ما مدیریت انبار خودمان است و دستورات از اداره مرکزی به ما ابلاغ می‌شوند.

نکته: برنامه به صورت interactive خط فرمان اجرا می‌شود. و تا وقتی که exit وارد نشود ادامه می‌یابد.

نکته: ورودی و خروجی دقیقا مطابق مستند باید باشند. (حتی در کوچک یا بزرگ بودن حروف)

نکته: مقادیری که درون {}آمده اند را به طور مفهومی بررسی و چاپ کنیم نه خود حروف کلمه:)

نکته: ترتیب بررسی خطاهای ممکن باید طبق ترتیب داک باشد!

نکته: در هر جایی از برنامه اگر ورودی کاربر با هیچکدام از قالب ورودی های زیر تطابق نداشته باشد باید خطای زیر چاپ شود:

```
[-] invalid command
```

راه اندازی ساختمان ذخیره‌سازی:

```
storage setup {m} {n} {default_capacity} {default_temperature}
```

توضیح: با اجرای این دستور یک محل ذخیره سازی ایجاد می شود. خروجی دستور: اگر محل ذخیره سازی این انبار از قبل وجود داشته باشد باید خطای زیر چاپ شود:

```
[-] storage is already setup
```

اگرظرفیت پیشفرض منفی باشد یا ابعاد ساختمان ذخیره سازی منفی باشد باید خطای زیر چاپ شود:

```
[-] storage setup failed
```

در غیر اینصورت باید ساختمان ذخیره سازی ایجاد شده و پیام زیر چاپ شود:

```
[+] storage setup successfully
```

ورود کالا:

از جمله رازهای بسیار مهم انبارداری، "جا کن" است.

انباردار باید بتواند بارها را تا حدی که می‌تواند در انبار قرار دهد. او میتواند پکیج ها را باز کرده و تقسیم کند. برای مثال، انباری با دو اتاق را در نظر بگیرید که اتاق اول فضایی به حجم ۳ و اتاق دوم فضایی به حجم ۵ خالی دارد. انباردار پکیجی شامل ۸ گوشی، که هر گوشی دارای حجم ۱ است را دریافت میکند. هیچکدام از اتاق ها ۸*۱ = ۸ جا ندارند! در نتیجه انباردار این پکیج را به دو پکیج ۳ تایی و ۵ تایی تبدیل می‌کند تا بتواند همه کالا ها را در انبار نگه‌داری کند.

بعد از دریافت دستور ورود کالا ، انبار دار از اولین اتاق، شروع به بررسی می‌کند (اتاقی به مختصات (۰،۰)). اگر جای خالی (حتی برای یکی از کالا های درون جعبه!) وجود داشت، هرچقدر که می‌تواند در آن جا کالا قرار می‌دهد. سپس با بقیه کالا ها و چرخ دستی خود، به اتاق بعدی انبار می‌رود. توجه شود که این پیمایش سطر به سطر بوده و این روند تا

زمانی ادامه دارد که تمام کالا ها در انبار جا بگیرند یا تمام خانه ها بررسی شوند.
در صورتی که فضای کافی برای همه کالا ها نبود، کالاهای باقیمانده خروجی داده می‌شوند .
نکته: کالا تنها در صورتی می تواند وارد یک اتاق با گنجایش کافی شود که با آن هم دما باشد! در غیر اینصورت باید به سراغ اتاق بعدی برویم! نکته ی مهم دیگر آن است که نمی توان یک کالا را برای جا کردن در یک اتاق تکه کرد ! برای مثال فرض کنید میخواهیم محتوای بسته ای شامل ۳ کالا ی ۴ لیتری را در اتاق های باقی مانده که تنها هر یک ۲ لیتر جا دارند بگذاریم . انباردار نمیتواند نصف هر کالا را در هر اتاق گذاشته و به آنها آسیب بزند!!

```
package add {name} {price} {temperature} {amount} {volume}
```

توضیح: با اجرای این دستور طبق قواعد گفته شده ، یک پکیج با ویژگی های داده شده به انبار اضافه می شود. خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

اگر کالاهای موجود در پکیج در اتاق های انبار جا نشوند باید خطای زیر چاپ شود:

```
[-] low capacity! we can’t add this part of package:
Package:
name: {name}
price: {price}$
temperature: {temperature}C
amount: {amount}
volume: {volume}L
```

که amount همان تعداد کالایی میباشد که به دلیل کمبود فضا نتوانستیم به انبار اضافه کند. در غیر اینصورت باید پکیج به انبار اضافه شده و پیام زیر چاپ شود:

```
[+] package added successfully
```

خروج کالا:

برای خارج کردن تعداد مشخصی کالا ، انبار دار از اولین اتاق شروع به بررسی می‌کند (اتاقی به مختصات (۰،۰)). اگر کالای مد نظر را در اتاق پیدا کرد، بیشترین تعدادی که می‌تواند را از آن برمیدارد . در صورت نیاز در باقی اتاق ها به جستجوی میزان باقیمانده کالا برای خروج می پردازد. توجه شود که نحوه ی پیمایش اتاق ها همانند **ورود کالا** سطر به سطر میباشد

در نهایت انباردار میزان کالایی که نتوانسته به دلیل کمبود موجودی از انبار خارج کند را خروجی می‌دهد .

```
package remove {name} {amount}
```

توضیح: با اجرای این دستور طبق قواعد گفته شده ، تعدادی از کالای مورد نظر با نام ذکر شده و به تعداد مشخص شده ، در قالب بسته ای از انبار حذف می شود. خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

اگر نتوانیم به تعداد مورد نظر از کالایی که از ما خواسته شده از انبار خارج کنیم و توانستیم تنها بخشی از آن را خارج کنیم چنین پیامی باید چاپ شود :

```
[-] low amount! we can’t remove this part of package:
Package:
name: {name}
price: {price}$
temperature: {temperature}C
amount: {amount}
volume: {volume}L
```

Amount همان تعداد کالایی میباشد که به دلیل کمبود موجودی نتوانستیم از انبار خارج کنیم . اگر از کالای مورد نظر هیچ مقداری در انبار موجود نبود پیام زیر چاپ میشود :

```
[-] package not found
```

در صورت موفقیت آمیز بودن خروج کالا باید پیام زیر چاپ شود :

```
[+] package removed successfully
```

دریافت اطلاعات کالا:

```
package get data {name}
```

توضیح: با اجرای این دستور اگر کالا در مختصاتی وجود داشت، مختصات و تعداد آن کالا در آن مختصات را خروجی دهد. نکته: خروجی می‌تواند چند خط باشد چون یک کالا صرفا در یک اتاق قرار ندارد.

خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

اگر پکیجی با نام وارد شده در انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] package not found
```

در غیر اینصورت باید اطلاعات پکیج موردنظر طبق قالب پیام زیر چاپ شود:

```
[+] package data:
```

و در ادامه ، مختصات هر یونیت که پکیج در آن موجود است را به همراه مقدار پکیج در آن یونیت در خطوط مجزا چاپ می شود :

```
(row, column) : amount
```

مرور وضعیت انبار:

```
storage overview
```

توضیح: با اجرای این دستورنام همه کالا ها و تعدادی که از آن در انبار وجود دارد را خروجی دهد. نکته: خروجی می‌تواند چندخط باشد زیرا محل ذخیره‌سازی می‌تواند کالاهای متفاوتی داشته باشد.

خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

در غیر اینصورت باید اطلاعات محل ذخیره سازی انبار طبق قالب پیام زیردر غیر اینصورت باید اطلاعات محل ذخیره سازی انبار نمایش داده شود به اینصورت که در خط اول پیام زیر چاپ شود:

```
[+] overview:
```

و در ادامه ، نام هر کالای موجود در انبار به همراه مجموع مقدار آن در کل انبار به صورت زیر در خطوط مجزا چاپ کنید :

```
{name} : {amount}
```

تعیین دمای اتاق:

```
unit set temperature {i} {j} {T}
```

توضیح: با اجرای این دستور دمای اتاق با مختصات (i،j) به T تغییر پیدا می‌کند.
نکته: اگر کالایی در این اتاق وجود داشت نمی‌توانید دمای اتاق را تغییر دهید. زیرا که کالا قطعا با دمای اتاق سازگار است و در صورت تغییر دمای اتاق، کالا آسیب می‌بیند.
خروجی دستور:
اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

اگر اتاقی با موقعیت مکانی وارد شده در محل ذخیره سازی وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] unit not found
```

اگر درون اتاق کالایی وجود داشته باشد باید خطای زیر چاپ شود:

```
[-] unit should be empty
```

در صورت موفقیت آمیز بودن این عملیات ، پیام زیر باید چاپ شود :

```
[+] unit temperature set successfully
```

نقشه‌ی فضاهای خالی:

```
storage free capacity map
```

توضیح: با اجرای این دستور نقشه انبار با توجه به گنجایش باقی مانده اتاق ها چاپ می شود. خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

در غیر اینصورت نقشه با الگوی زیر چاپ می شود: خروجی یک جدول مستطیلی (آرایه ی دو بعدی) از اعداد است که خانه (۰، ۰) سمت چپ بالا قرار دارد. در هر خانه از جدول مستطیلی مقدار فضای خالی باقی مانده از unit متناظر چاپ میشود (عدد صحیح و نامنفی).

```
[+] free map:
{free capacity map}
```

نقشه ی حرارتی:

```
storage temperature map
```

توضیح: با اجرای این دستور نقشه انبار با توجه به دمای اتاق ها چاپ می شود. خروجی دستور: اگر محل ذخیره سازی انبار وجود نداشته باشد باید خطای زیر چاپ شود:

```
[-] storage is not setup
```

در غیر اینصورت نقشه با الگوی زیر چاپ می شود: خروجی یک جدول مستطیلی (آرایه ی دو بعدی) از اعداد است که خانه (۰، ۰) سمت چپ بالا قرار دارد. هر اتاق با مقدار دمایی که دارد نمایش داده می‌شود.

```
[+] temperature map:
{temperature map}
```

ورودی نمونه ۱

```
storage setup 5 4 5 5
package add a 100 5 5 5
storage overview
package get data a
unit set temperature 0 0 10
unit set temperature 5 4 10
unit set temperature 4 3 10
storage temperature map
storage free capacity map
package add n 100 5 3 6
package remove a 30
unit set temperature 0 0 5
package add m 10 5 1 5
package get data n
storage free capacity map
unit set temperature 3 3 3
storage temperature map
exit
```

خروجی نمونه ۱

```
[+] storage setup successfully
package added successfully
overview:
a : 5
package data:
(0, 0) : 1
(0, 1) : 1
(0, 2) : 1
(0, 3) : 1
(1, 0) : 1
unit should be empty
unit not found
unit temperature set successfully
temperature map:
5 5 5 5
```

```
5 5 5 5
5 5 5 5
5 5 5 5
5 5 5 10
free map:
0 0 0 0
0 5 5 5
5 5 5 5
5 5 5 5
5 5 5 5
low capacity! we can’t add this part of package:
Package:
name: n
price: 100$
temperature: 5C
amount: 3
volume: 6L
low amount! we can’t remove this part of package:
Package:
name: a
price: 100$
temperature: 5C
amount: 25
volume: 5L
unit temperature set successfully
package added successfully
package not found
free map:
0 5 5 5
5 5 5 5
5 5 5 5
5 5 5 5
5 5 5 5
unit temperature set successfully
temperature map:
5 5 5 5
5 5 5 5
5 5 5 5
5 5 5 3
```

```
5 5 5 10
```