



دانشکده مهندسی برق
دانشگاه صنعتی شریف

درس برنامه سازی شے گرا

نیمسال دوم ۱۴۰۲-۱۴۰۳
دکتر وثوقی وحدت، دکتر هاشمی

بارم

تمرین سوم

۰/۵۵

ا. کد حساب

۰/۷۰

Git .۲

۰/۵۰

Operating System .۳

مهلت تحویل:

جمعه ۵ اردیبهشت ۱۴۰۲

توجه:

این تمرین در مجموع دارای ۷۵/۱ نمره می‌باشد؛ ۲۵/۱ نمره بارم اصلی تمرین سوم، ۲۵/۰ امتیازی تمرین سوم و ۲۵/۰ دیگر می‌تواند امتیازی تمرین را جبران کند!

کد حساب

در این سوال قرار است شما برنامه‌ای بنویسید که اعمال اصلی حساب (ضرب ، تقسیم ، جمع و تفریق) و همچنین مقایسه دو ورودی را انجام دهد. توجه کنید که عملیات مورد نظر در ورودی به برنامه داده می‌شود و شما باید پاسخ را در خط بعدی چاپ کنید.

توجه کنید که اعداد و کاراکتر های شما در محدوده Integer ، Long ، Double، همچنین کاراکتر های موجود در جدول اسکی می‌باشند و شما باید پاسخ را در همان محدوده ورودی‌ها چاپ کنید.

تضمين می‌شود هر دو ورودی از یک جنس باشند (مثلا شما دو ورودی Double دریافت می‌کنید ، و در پاسخ نیز باید پاسخ چاپ شده از همین جنس باشد).

توجه: برای حل سوال باید از generic استفاده کنید.(به قول معروف اجباریه دیگه، فقط به این نمره میدیم!)

عمل جمع

$$5 + 6$$

$$11$$

شما به ترتیب 5 و 6 را دریافت کرده اید و با تشخیص عملیات جمع پاسخ 11 را که از نوع Integer می‌باشد را به همان جنس ورودی‌ها چاپ کرده اید.

عمل تفریق

$$2.0 - 2.0$$

$$0.0$$

عمل ضرب

`60 * 10
600`

#عمل تقسیم

`6 / 5
1`

در اینجا بدلیل اینکه ورودی‌های داده شده از نوع Integer است، خروجی داده شده نیز از همان نوع بوده و حاصل برابر 1 می‌باشد.

`6.0 / 5.0
1.2`

`6000000000000000 / 5000000000000000
1`

عمل مقایسه

این عملیات با دریافت حرف C به عنوان عمل مقایسه گر از بین دو ورودی، ورودی بزرگتر را چاپ می‌کند.

`a < b`

چون ورودی‌ها از نوع کاراکتر هستند، در خروجی کاراکتری که در جدول اسکی دارای مقدار (عدد) بیشتری است چاپ می‌شود. در نهایت نیز با دریافت دستور end برنامه تمام می‌شود.

`b`

Git

در این سوال ما قصد داریم تا نمونه‌ای ساده سازی شده ای از Git را پیاده سازی کنیم. Git که احتمالاً در پروژه با آن خیلی سر و کار خواهد داشت، یک نرم افزار کنترل نسخه است و می‌توان با استفاده از آن به تاریخچه کامل تغییرات یک پروژه دسترسی یافت و آنها را بازنگری کرد. Git تغییرات یک پروژه را ضبط کرده و در یک دیتابیس ذخیره می‌کند. ما می‌توانیم این تغییرات را مشاهده کنیم و ببینیم که چه کسی چه تغییری را و در چه زمانی ایجاد کرده است. این از این نظر مفید است که می‌توانیم هرگاه بخشی از پروژه به مشکل برسورد و یا تغییرات نامطلوبی اتفاق افتاد، آن را به حالت قبلی برگردانیم و پروژه قبلی را بازیابی کنیم.

روشی که با استفاده از گیت می‌توان snapshot هایی از پروژه را ذخیره کرد به این صورت است که ابتدا فایل‌ها به محیطی به نام staging area اضافه می‌شوند سپس یک نسخه از commit کردن می‌گویند و بعدها می‌توان پروژه را به این تاریخچه پروژه می‌رود. به این نسخه گرفتن commit کردن می‌گویند و بعدها می‌توان پروژه را به این commit ها برگرداند. اساس کار این سوال نیز بر همین مفهوم است.

اکثر دستورات خواسته شده در سوال زیر نسخه هایی ساده شده از دستورات git هستند.

(۱) ایجاد مخزن ▼

این دستور یک repository با اسم داده شده می‌سازد. توجه کنید که اسم repository داخل " " قرار ندارد.

```
new repository [repository name]
```

اگر از قبل rep ای با اسم داده شده وجود دارد باید خط زیر چاپ شود .

```
Repository with this name already exists
```

اگر دستور موفقیت آمیز بود، باید خط زیر چاپ شود

Repository created successfully!

۲) گشودن مخزن ▼

این دستور repository با اسم داده شده را باز میکند، به این معنا که از بعد از این دستور تا دستور مشابه بعدی، همه فعالیت ها در repository باز شده اتفاق می افتد. توجه کنید که اسم repository داخل " " قرار ندارد.

open repository [repository name] .

اگر rep ای با این اسم وجود نداشته باشد باید خط زیر چاپ شود

No repository exists with this name

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

Repository [repository name] opened successfully!

۳) ساخت فایل ▼

این دستور فایل جدیدی با اسم و محتوای داده شده در rep باز شده فعلی میسازد.

new file ["file name"] ["file content"]

اگر از قبل فایلی با این اسم در rep باز فعلی وجود دارد باید خط زیر چاپ شود

File with this name already exists in this Repository

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

File created successfully!

۴) ویرایش فایل ▼

این دستور محتوای فایل با اسم داده شده را با محتوای جدید جایگذاری می‌کند.

```
edit file ["file name"] ["new file content"]
```

اگر فایلی با این اسم در rep باز فعلی وجود ندارد باید خط زیر چاپ شود

No file exists with the given name

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

Edited successfully!

۵) افزودن همه ▼

این دستور تمام فایل‌های rep باز فعلی را به staging area اضافه می‌کند.

```
add -all
```

بعد از این دستور باید خط زیر چاپ شود

Added

۶) افزودن به فضای ذخیره سازی ▼

این دستور فقط فایل با اسم داده شده را به staging area اضافه می‌کند.

```
add ["file name"]
```

اگر فایلی با این اسم در rep باز فعلی وجود ندارد باید خط زیر چاپ شود

```
No file exists with the given name
```

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

```
Added
```

▼ ۷) ثبت کامیت

این دستور یک commit با عنوان داده شده می سازد.

```
commit ["commit title"]
```

اگر فایلی در staging area وجود ندارد باید خط زیر چاپ شود

```
No files are added
```

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

```
Committed
```

تضمین می شود که اسم commit ها تکراری نخواهد بود.

▼ ۸) حذف همه از فضای ذخیره سازی

این دستور تمام فایل های staging area را از staging area حذف می کند.

```
un-stage all
```

بعد از این دستور باید خط زیر چاپ شود

Removed from stage

▼ ۹) حذف از فضای ذخیره سازی

این دستور فقط فایل با اسم داده شده را از staging area حذف می‌کند.

un-stage ["file name"]

اگر فایلی با این اسم در staging area وجود ندارد باید خط زیر چاپ شود

No file exists with the given name

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

Removed from stage

▼ ۱۰) حذف از مخزن

این دستور فایل با اسم داده شده را از rep باز فعلی حذف می‌کند.

remove ["file name"]

اگر فایلی با این اسم در rep باز فعلی وجود ندارد باید خط زیر چاپ شود

No file exists with the given name

اگر دستور موفقیت آمیز بود باید خط زیر چاپ شود

Removed

▼ ۱۱) بازیابی 1

این دستور تمامی فایل هایی از rep فعلی که در commit مشخص شده وجود دارند را به حالتشان در زمان ایجاد این commit برمیگرداند. در واقع محتوای این فایل ها به محتوای قبلیشان در زمان ایجاد این commit عوض میشود. commit مورد نظر به دو روش مشخص میشود. یا عنوان commit یا شماره commit داده میشود.

توجه کنید که اگر اسم commit داده شود، این اسم در " " قرار دارد اما اگر شماره داده شود در " " قرار ندارد.

```
restore ["commit title" / number of commit]
```

اگر commit ای با اسم داده شده وجود نداشته باشد باید خط زیر چاپ شود

No commit exists with the given name

اگر commit ای با شماره داده شده وجود نداشته باشد باید خط زیر چاپ شود

No commit exists with the given number

در صورت موفقیت آمیز بودن دستور خط زیر چاپ شود

Restored

▼ ۱۲) بازیابی 2

این دستور مانند قبلی است. با این تفاوت که فقط فایل با نام داده شده به حالت قبل بر میگردد.

```
restore ["file name"] ["commit title" / number of commit]
```

اکسپشن های این دستور نیز مانند دستور قبل است. فقط در صورت وجود نداشتن فایل با اسم داده شده در commit داده شده باید خط زیر چاپ شود

No file exists in the given commit with the given name

توجه کنید که الوبیت چاپ اکسپشن با اکسپشن وجود نداشتن commit است.

در این دستور نیز در صورت موفقیت آمیز بودن دستور خط زیر چاپ می‌شود.

Restored

▼ مقایسه (۱۳)

در این دستور یک فایل داده می‌شود که باید بین دو نسخه متفاوت آن که که توسط دو مشخص می‌شود مقایسه صورت گیرد. مقایسه به این صورت است که محتوای دو فایل کلمه به کلمه مقایسه شده و چاپ می‌شوند و کلماتی که در دو نسخه متفاوت هستند باید با دو * در دو طرفشان مشخص شوند. در این دستور هر دو commit یا با عدد یا با عنوان داده می‌شوند.

```
diff ["file name"] ["1st commit title" / 1st commit number] ["2nd commit title" / 2nd
```

برای مثال

```
*Hello* world
```

```
*Goodbye* world
```

البته اگر یکی از کلمات متفاوت به طور دست نخورده در فایل دیگر وجود دارد ولی جابجا شده نباید با ستاره مشخص شود. برای مثال

```
Hello world
```

```
Hello *beautiful* world
```

همچنین اگر کلمه‌ای اضافه شده باشد نیز باید با ستاره مشخص شود

```
Hello world
```

```
Hello world *I'm* *ready*
```

توجه کنید که کدتان باید case sensitive باشد

```
Hello *world*
```

```
Hello *World*
```

همچنین توجه کنید که دو نسخه چاپ شده با یک new line از هم جدا شده‌اند.

در صورت وجود نداشتن یکی از commit‌ها خط زیر چاپ شود

```
One of the commits doesn't exist
```

در صورت وجود نداشتن فایل در یکی از commit‌ها خط زیر چاپ شود

```
The given file doesn't exist in one/both of the given commits or its removed
```

▼ ۱۴) نمایش محتوای فایل

محتوای فایل مشخص شده را نمایش می‌دهد.

```
show ["file name"]
```

اگر فایل با اسم داده شده در rep باز فعلی وجود نداشت خط زیر چاپ شود

```
No file exists with the given name
```

▼ ۱۵) نمایش وضعیت

این دستور تمامی فایل های rep باز فعلی را به نمایش می‌گذارد و وضعیت staged یا not staged آنها را نیز مشخص می‌کند.

status

برای مثال

```
file1 staged
file2 staged
file3 not staged
```

ترتیب نمایش فایل ها بر حسب زمان به وجود آمدن آنهاست. به طوری که جدیدترین ها در پایین قرار می‌گیرند

اگر فایلی در rep نباشد خط زیر چاپ می‌شود

There are no files in the repository

▼ ۱۶) نمایش سابقه

لیستی از تمامی commit های انجام شده به نمایش می‌گذارد.

log

مثال

1. Commit1
2. Commit2

توجه کنید که بین نقطه و عنوان commit یه فاصله وجود دارد.

اگر تا الان commit ای انجام نشده است باید خط زیر چاپ شود

```
No commits have been done yet
```

۱۷) پایان ▼

با وارد کردن آن برنامه بدون هیچ خروجی به اتمام می‌رسد.

```
end
```

نمونه‌ها ▼

ورودی نمونه ۱

```
new repository x
open repository y
open repository x
new file "file1" "content1"
new file "file2" "content2"
add -all
un-stage "file2"
status
commit "first commit"
log
show "file2"
edit file "file1" "new content1"
show "file1"
restore "file3" 1
restore "file1" "first commit"
show "file1"
end
```

خروجی نمونه ۱

Operating System

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 آن ها خواهیم رفت.

▼ انواع فایل

در ابتدا به نکات کلی سوال توجه کنید: در این سؤال با ۳ نوع فرمت فایل سروکار داریم عکس (img)، ویدئو (mp4) و متن (txt) هستند.

تمام فایل‌ها یک سری ویژگی‌های مشترک دارند مانند؛ آدرس، اسم و... که در ادامه با دستورات مربوط به آن ها آشنا می‌شویم. اما بعضی ویژگی‌ها هستند که بنابر نوع فایل متفاوت‌اند.

این ویژگی‌ها به ترتیب برای هر فایل به شرح زیر هستند:

ویدیو:

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

طول یا همان مدت زمان ویدیو به فرمت hh:mm:ss داده می‌شود.

متن:

Text

نگران نباشین! قرار نیست با فایل متن واقعی کار کنیم، صرفاً کافیه یه رشته در نظر بگیرین

عکس:

Resolution

رزولوشن یا وضوح تصویر به صورت دو عدد با علامت * در بینشون مثل 1536*2048

Extension(jpg, png)

▼ نکات

نکته ۰: هر فolder از تعدادی فایل و فولدر دیگر تشکیل شده است.

نکته ۱: 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 ۲: دقت کنید که ارورهایی که برای هر دستور در ادامه گفته شده را به ترتیب سوال چک کنید.

نکته 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 است، در ادامه درباره این بیشتر توضیح داده شده است)

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

نکته ۵: از space های اضافه 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 کوچک و بزرگ بودن حروف دستورات و ارور های که باید چاپ کنید، دقت کنید):

▼ 1. Install OS

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 می شوند 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 پوش کنید) این دستور تضمین می شود که فقط یک بار وارد شود. (البته الزاماً دستوری که بلا فاصله بعد دستور بالا وارد می شود، این دستور نیست و باید معتبر بودن آن را چک کنید)

```
#hardSize #drivesNum
```

سپس در #drivesName هر خط یک حرف بزرگ انگلیس داده می شود (AZ-) که نام هر درایو را نشان می دهد سپس با یک فاصله حجم مورد نظر برای آن درایو وارد می شود. (فقط عدد و بدون MB است و حتی می تواند صفر باشد) توجه کنید که نباید نام درایو ها تکراری باشند.

توجه: اولین درایوی که وارد می شود و معتبر باشد، به عنوان درایو سیستم در نظر گرفته می شود 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 کarakترهای دیگری غیر از حروف بزرگ بود باید ارور زیر چاپ شود:

```
invalid name
```

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

```
insufficient hard size
```

تضمين می شود 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 می یابد)

حال که سیستم عامل با موفقیت نصب شد وقتی که شروع کنیم به ساخت فایل و فolder و کار کردن با اون ها؛

▼ 2.Open

```
open #folderName
```

با این دستور وارد فolder با اسم داده شده می شوید (در صورت معتبر بودن دستور، بازکردن فolder به این معنی است که مکان فعل تغییر می کند و وارد آن فolder می شوید)

توجه: این دستور برای فایل ها وجود ندارد و حتی در صورت که فایل با اسم داده شده نیز وجود داشته باشد اجرا نمی شود.

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

```
invalid name
```

▼ 3.Go to

این دستور در هر کجای برنامه وارد شود بایست وارد درایو با نام داده شده شوید به این معنی که مکان فعل تغییر می کند و وارد آن درایو می شوید.

```
go to drive #driveName
```

اگر درایوی با نام داده شده وجود نداشت باید ارور زیر چاپ شود

```
invalid name
```

▼ 4.Back

```
back
```

با این دستور یک مرحله به عقب بر می گردید.

#مثال 1: اگر مکان فعلی

```
H:\Spirng\HW3
```

باشد و دستور back وارد شود، مکان به

```
H:\Spirng
```

تغییر می کند. #مثال 2: اگر مکان فعلی باشد H:\ و دستور back وارد شود، مکان شما تغییر نمی کند و اروری نیز چاپ نمی شود.

▼ 5.Create Folder

```
create folder #name
```

با دستور بالا فolderی با اسم داده شده در مکان فعلی ساخته می شود. نام فولدر می تواند 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 فolder عبارت زیر چاپ شود:

```
folder created
```

▼ 6.Create File

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

با دستور بالا فایل با اسم داده شده در مکان فعل ساخته می شود. سایز داده شده بحسب MB است. نام فایل می تواند 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 txt، img، mp4 نبود باید ارور زیر چاپ شود:

```
invalid format
```

اگر حجم فایل داده شده + حجم فایل های 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 ای از ۳ نوع txt، img، mp4 است که بسته به اینکه کدام نوع باشد باید فایل از نوع متن، عکس یا ویدیو بسازید. سپس اگر ورودی مشکل نداشت، به نسبت فرمت فایل، معیارها را دریافت می کند. به این صورت که به ترتیب آورده شده در بالا، معیارها چاپ می شود و مقادیر مورد نظر از ورودی دریافت می شود.

مثلا اگر ویدیو بود باید چاپ کنید:

Quality:

سپس ورودی بگیرید و سپس چاپ کنید:

Video Length:

و ورودی بگیرید.(تصمین می شود که این ورودی ها معتبر باشند)

در نهایت اگر دستور معتبر بود پس از گرفتن ویژگی های فایل که در بالا گفته شد، و پس از ساخت فایل

عبارت زیر چاپ شود:

file created

▼ 7.Delete File

delete file #fileName

با این دستور فایل با اسم داده را حذف می کنید و حجم آن را نیز از حجم استفاده شده ازفضای درایو حذف می کنید.

اگر فایل 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

▼ 8.Delete Folder

```
delete folder #folderName
```

با این دستور فolder را با اسم داده را حذف می کنید. وقت کنید که با حذف فolder تمام فایل ها و فولدرهای داخل آن نیز حذف خواهند شد.

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

```
invalid name
```

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

```
folder deleted
```

▼ 9.Rename File

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

اسم فایل با اسم داده شده را تغییر م دهید. اگر فایل با نام داده شده در آن م ان 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
```

بود پس از تغییر نام فایل عبارت زیر چاپ شود:

```
file renamed
```

▼ 10.Rename Folder

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

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

```
invalid name
```

اگر فolder 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 فolder عبارت زیر چاپ شود:

```
folder renamed
```

▼ 11.Status

```
status
```

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

```
H:\Java Projects\Spirng\HW3
```

سپس نام تمام فایل ها و فولدرهای آن م ان را به ترتیب نام و به فرمت زیر چاپ کنید(به ترتیب ابتدا فولدرها و سپس عکس ها، متن ها، ویدیوها):

هر خط شامل یک اسم، فاصله، نوع آن فایل یا فolder، فاصله، حجم آن فایل یا فolder به عنوان مثال:

Folders:

Q2 50MB

q1 15MB

Files:

x img 2MB

a txt 80MB

B mp4 400MB

▼ 12.Drive Status

```
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 کل آن درایو، فاصله، حجم اشغال شده آن درایو توجه:

ترتیب درایو ها همان ترتیب ساخته شدن آن ها باید باشد.

به عنوان مثال:

C 75000MB 48000MB

H 100000MB 500MB

▼ 13.File Status

```
print file stats #fileName
```

مشخصات یک فایل را به فرمت زیر چاپ می کند.(در صورت معتبر بودن دستور)

:۱##مثال

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

:۲##مثال

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

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

```
invalid name
```

▼ 14. Write Text

```
write text #textFileName
```

باید در فایل متن با نام داده شده، متن که داده می شود (بلافاصله در دستور بعدی) را بنویسید (متن قبل موجود در آن پاک می شود). توجه کنید که متن داده شده می تواند شامل فاصله و کاراکترهای غیر حرف نیز باشد. فرض کنید که حجم فایل تغییر نمی کند. اگر فایل با نام داده شده در آن مکان وجود نداشت باید ارور زیر چاپ شود:

```
invalid name
```

اگر فایل با نام داده شده در آن مکان وجود داشت اما فایل متن 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 متن خواهد بود ولی اگر معتبر نباشد دستور بعدی آن نشانگر متن نیست.

▼ 15.Frequent Folders

```
print frequent folders
```

این دستور در هر جایی از برنامه می تواند وارد شود و باید آدرس و تعداد دفعات بازکردن ۵ فolder را چاپ کنید که بیشتر از دیگر فولدرها برای آن ها دستور open را انجام داده اید.(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 ورودی خواهد بود. اگر تعداد ورود برای دو فولدر یکسان بود به ترتیب حروف الفبا چاپ کنید.(دقت کنید که باید آدرس آن ها را از لحاظ حروف الفبا مقایسه کنید)

مثال:

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

▼ 16.OS Information

```
print OS information
```

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

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

▼ 17.End

برنامه در نهایت با دستور زیر به پایان می رسد

```
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
```

▼ نمونه‌ها

1 نمونه رویدی

```
install OS sut sut1401
1000 3
C 500
D 300
F 200
go to drive D
create folder folder1
create folder folder1
create folder Folder1
go to drive C
create folder folder1
create folder folder2
open folder
open folder1
open folder2
go to drive D
```