به نام خدا

پروژه پایانی درس مبانی برنامه‌نویسی

یک زبان اسکریپتی بسازید

آخرین زمان تحویل پروژه: ۳ بهمن

مجموع نمره: ۱۰۰

پروژه باید در سامانه HWS بارگزاری شود و سپس در یک جلسه پرسش و پاسخ آنلاین (با دوربین) باید ارائه داده شود. زمان ارائه طبق اطلاع بعدی مشخص می‌شود و توجه کنید که پروژه بدون ارائه، نمره‌ای نخواهد داشت.

مقدمه

زبان های اسکریپتی زبان هایی هستند که باید توسط یک برنامه خوانده، تفسیر و اجرا شوند. خود پایتون یک زبان اسکریپتی است؛ کدی که شما به زبان پایتون می‌نویسید توسط مفسر پایتون خوانده، تفسیر و اجرا می‌گردد. حال از شما خواسته شده تا یک زبان اسکریپتی شخصی سازی بسازید. این زبان را 4001 می‌نامیم!

سعی کنید هر بخشی از این پروژه را که می‌توانید انجام دهید تا نمره آن بخش را بگیرید. نکته دیگر این که شماره خط هایی که در متون کد این مقاله هستند صرفا برای نمایش کد هستند و به هیچ وجه نیازی به پیاده سازی آن در برنامه شما نیست.

دو حالت اجرای برنامه

برنامه شما باید دو حالت داشته باشد. در یک حالت کاربر خط به خط دستورات را وارد می‌کند. در این حالت برنامه شما یک علامت '>' (علامت بزرگ تر از) نمایش می‌دهد که به معنی آن است که آماده دریافت دستور از کاربر است و پس از اجرای دستور نتیجه را به کاربر نشان می‌دهد. و در حالت دیگر فایلی را به برنامه می‌دهد و اعمالی که خط به خط در فایل نوشته شده اجرا می‌شوند و نتایج یکی پس از دیگری بر صفحه ظاهر می‌شوند. تمامی دستورات و شیوه گرفتن فایل از کاربر در بخش های بعدی توضیح داده شده.

شیوه نوشتار زبان ۴۰۰۱

شیوه نوشتار (syntax) همان اصول و قواعد ساده نوشتن کد است. زبان ۴۰۰۱ یک زبان تابع محور است و برای هر دستور یک تابع فراخوانی می‌شود و بسته به نیاز دستور پارامتر های مختلفی به آن ارسال می‌شوند.

بنابراین هر خط یک دستور و هر دستور شامل دو بخش کلی است. این دو بخش نام تابع و پارامتر های آن هستند که با فاصله از هم جدا می‌شوند. برای مثال دستور زیر تابع echo را برای نمایش جمله hello world فراخوانی می‌کند:

```
01 echo hello world
```

در این مثال تابع echo تعداد متغیری پارامتر بسته به نیاز کاربر دریافت می‌کند، مثلا در این مورد دو پارامتر hello و world است که با فاصله از هم جدا شده اند. درواقع تابع echo دارای خاصیتی است که در هنگام نمایش خروجی بین هر تعداد پارامتر که داده شود یک کاراکتر فاصله بگذارد تا جمله به درستی نمایش داده شود.

متغیر ها

بارم: ۲۰ نمره

اگر برنامه شما از متغیر هم پشتیبانی کند و طبق شرایطی که در توابع گفته شده بتوان از متغیر ها استفاده کرد، نمره شما از ۱۰۰ نمره کل محاسبه می‌شود. در غیر این صورت نمره شما از ۶۰ نمره باقی مانده محاسبه می‌شود. در زبان 4001 متغیر ها می‌توانند اعداد را در خود ذخیره کنند (در دستور calc گفته شده که این عدد به اختیار خودتان و به صورت امتیازی می‌تواند عدد صحیح و یا اعشاری باشد). نام متغیر ها می‌توانند شامل حروف بزرگ و کوچک و عدد باشد. تمام علائم به جز علامت '_' (علامت آندرلاین یا آندراسکور) غیرمجاز هستند. درهنگام استفاده از یک متغیر در کد ها در دوطرف آن علامت % می‌ماند. مثلا اگر متغیری با نام test_2 داشتیم که مقدار ۷ در آن ذخیره شده بود آنگاه خواهیم داشت:

```
01 |> echo value is %test_2%
02 |value is 7
```

در مثال بالا ابتدا متن value is نمایش داده شده و سپس عبارت %test_2% با مقدار این متغیر جایگزین شده.

توابع

همان طور که پیش تر گفته شد هر خط دستور با نام یک تابع آغاز می‌شود. برای نمایش شیوه نوشتار هر دستور از علامت < > برای توضیح پارامتر استفاده شده.

تابع echo

نمره: ۲

دستور echo برای نمایش جملات و یا مقادیر به کار برده می‌شوند

```
01 |> echo <sentences parts>
```

این تابع بی‌شمار پارامتر را به عنوان تکه های جمله دریافت می‌کند و نمایش می‌دهد.

مثال:

```
01 |> echo hello this is an example
02 |hello this is an example
```

اگر برنامه شما از متغیر پشتیبانی می‌کند این تابع باید بتواند متغیر ها را نیز نمایش دهد. برای مثال اگر متغیر test برابر با ۷ باشد آنگاه:

```
01 |> echo test is %test%
02 |test is 7
```

تابع pause

نمره: ۲

```
> pause
```

این تابع هیچ گونه پارامتری دریافت نمی‌کند و جهت ایجاد وقفه در برنامه؛ عموما بعد از نمایش یک نتیجه به کاربر مورد استفاده قرار می‌گیرد. زمانی که این تابع فراخوانی می‌شود برنامه باید یک پیغام خاص و ثابت به کاربر نمایش دهد و از او بخواهد تا کلید Enter را جهت ادامه فرایند فشار دهد. این پیغام در مثال زیر آمده:

```
> pause
Please press ENTER to continue...
```

نکته مهم: از آنجایی که پایتون امکان دریافت یک تک کلید را ندارد ممکن است کاربر کلید های دیگری را نیز بزند و سپس کلید enter را فشار دهد یا مثلا با اینکار یک جمله را به برنامه شما ارسال کند. این موارد نیازی به بررسی ندارند و اهمیتی ندارند. یعنی برای مثال ممکن است چنین حالاتی رخ دهند که در این پروژه مهم نیست:

```
> pause
Please press ENTER to continue...user input before enter
```

متنی که با رنگ سبز مشخص شده ورودی کاربر قبل از کلید enter است. تنها کلیدی که موثر است کلید enter است.

تابع calc

نمره: ۱۴

این تابع که برای محاسبات ریاضیاتی استفاده می‌شود و باید بتواند یک عبارت ریاضیاتی از نوع prefix را محاسبه کند. درمورد عبارات ریاضی prefix؛ فرق آن با عبارات معمول از نوع infix و چگونگی محاسبه و تبدیل آن مفصلاً در کلاس حل تمرین صحبت شد پس از ارائه توضیحات در این خصوص در صرف‌نظر می‌شود. از آنجا که اجزای عبارات prefix با فاصله از هم جدا می‌شوند پس در پارمتر های جدا به این تابع ارسال می‌شوند.

اعمال ریاضیاتی شامل چهار عمل اصلی یعنی + ، - ، * ، / (جمع، تفریق، ضرب و تقسیم) و همچنین عملگر ^ برای توان و # برای باقی‌مانده (پیمانه) و \ برای تقسیم صحیح می‌شود. (در مجموع ۷ عملگر)

توجه کنید که ورودی باید کنترل شود. عملگر های اشتباه یا وارد کردن حروف به جای اعداد باید به کاربر هشدار داده شوند. هم‌چنین دقت کنید که اعداد می‌توانند علامت مثبت و منفی نیز داشته باشند.

اگر برنامه شما از متغیر پشتیبانی می‌کند عملکرد مورد انتظار به این صورت است:

```
> calc <variable name> <expression>
```

در این حالت پارامتر اول نام متغیری است که حاصل باید در آن ذخیره شود و پارامتر های بعدی بخش های عبارت هستند. این تابع پس از محاسبه عبارت، نتیجه را در متغیر ذخیره می‌کند و چیزی نمایش داده نمی‌شود.

مثال:

```
> calc test + 1 2
> echo %test%
3
```

متغیر ها می‌توانند در عبارت ریاضی نیز استفاده شوند. در ادامه مثال بالا داریم:

```
> calc test * %test% 2
> echo %test%
6
```

از این تابع می‌توان برای ذخیره یک مقدار عددی در یک متغیر نیز استفاده کرد. در مثال زیر مقدار متغیر test برابر با ۵ شده:

```
> calc test 5
```

اما در صورتی که برنامه شما از متغیر پشتیبانی نمی‌کند عملکرد این تابع به این صورت است:

تمام پارامتر های تابع بخش های عبارت ریاضی هستند و پس از محاسبه عبارت حاصل آن به کاربر نمایش داده می‌شود.

مثال:

```
> calc + - * 2 3 4 5
7
```

امتیازی (۵ نمره): دستور calc اگر بتواند از اعداد اعشاری مثبت و منفی پشتیبانی کند (و در عین حال ورودی کنترل شود) نمره امتیازی به شما تعلق می‌گیرد.

تابع get

نمره: ۴

```
> get <variable name> <prompt>
```

اگر برنامه شما از متغیر پشتیبانی می‌کند باید دارای این تابع باشد. این تابع یک پیغام را به کاربر نمایش می‌دهد و یک مقدار عددی (که می‌تواند مانند دستور calc اعشاری یا صحیح باشد) را از کاربر می‌گیرد و در متغیر ذخیره می‌کند. توجه کنید که این تابع باید کنترل ورودی نیز داشته باشد. هم‌چنین اعداد می‌توانند مثبت و منفی باشند.

تابع prims

نمره: ۱۲

```
> prims <n>
```

این تابع جهت نمایش تمام اعداد اول از ۱ تا خود n است و هر عدد با علامت , جدا می‌شود:

```
> prims 7
2, 3, 5, 7
```

درصورت پشتیبانی از متغیر ها این تابع نیز باید بتواند یک متغیر را به عنوان پارامتر دریافت کند. مثال:

```
> calc b 5
> prims %b%
2, 3, 5
```

تابع do

نمره: ۱۸

```
> do <n> <command>
> do <n>
>     <command 1>
>     <command 2>
>         ...
```

دستور do یک حلقه تکرار است. این حلقه یک یا چند دستور را n بار اجرا می‌کند. شیوه نوشتن حلقه به دو صورت است. در شیوه اول برای اجرای یک دستور، می‌توان دستور را در پارامتر دوم تابع do نوشت (متن دستور می‌تواند دارای فاصله نیز باشد). در حالت دوم برای اجرای چند دستور، ابتدا حلقه تعریف و سپس اجرا می‌شود. در این حالت در خط اول تابع do و تعداد دفعات تکرار (n) نوشته می‌شود. سپس برنامه شما یک کاراکتر ':' (نقطه بیانی) به کاربر نمایش می‌دهد که به معنای آن است که منتظر دریافت دستورات است. سپس هر دستور در یک خط وارد می‌شود تا زمانی که یک خط خالی بماند که به معنی پایان حلقه است و حلقه باید اجرا شود. در صورتی که دستور وارد شده نامعتبر باشد به کاربر پیغام خطا نمایش داده می‌شود و حلقه منتفی شود.

مثال برای یک دستور:

```
> do 3 echo hello
hello
hello
hello
```

مثال برای چند دستور:

```
> do 3
: calc + 2 3
: prims 9
:
5
2, 3, 5, 7
5
2, 3, 5, 7
5
2, 3, 5, 7
```

درصورتی که برنامه شما از متغیر ها پشتیبانی می‌کند، تابع do می‌تواند تعداد دفعات تکرار را از متغیر دریافت کند. همچنین دستور یا دستورات نیز می‌توانند مانند حالت عادی شامل متغیر هایی باشند. مثال:

```
> get n n=
n=5
> calc a 0
> do %n%
: calc a + %a% 1
: echo %a%
:
1
2
3
4
5
```

مثال ۲:

```
> calc i 3
> calc ii 4
> do %i% echo %ii%
4
4
4
```

تابع open و close

نمره: ۱۰

```
> open <file>
> close
```

دستور open باعث می‌شود تا برنامه شما تمام خروجی های پس از آن را در فایلی که کاربر مشخص کرده تا زمانی که کاربر دستور close را وارد نکرده ذخیره کند؛ یعنی با اجرای دستور close دیگر خروجی ها در فایل ذخیره نمی‌شوند. توجه کنید که نام فایل می‌تواند حاوی حرف فاصله نیز باشد. پس از اجرای دستور open و close به ترتیب پیغام های زیر باید نمایش داده شوند. در هر دوی این پیغام ها <file> نام فایلی است که کاربر مشخص کرده.

```
File "<file>" opened
File "<file>" closed
```

مثال:

```
> open test file.txt
File "test file.txt" opened
> echo hello
hello
> prims 10
2, 3, 5, 7
> calc + + + + 1 2 3 4 5
15
> echo end
end
> close
File "test file.txt" closed
```

محتویات فایل test file.txt به این صورت است:

```
hello
2, 3, 5, 7
15
end
```

دقت کنید که خطوط خالی در فایل نوشته نمی‌شوند.

اگر برنامه شما از متغیر پشتیبانی می‌کند بنابراین کل یا بخشی از نام فایل می‌تواند متغیر باشد. مثال:

```
> calc i 0
> do 5
: calc i + %i% 1
: open test%i%.txt
: echo hi file %i%
: close
:
hi file 1
hi file 2
hi file 3
hi file 4
hi file 5
```

در نتیجه اجرای کد بالا پنج فایل با نام های test1.txt تا test5.txt ساخته می‌شود

تابع read

نمره: ۸

این تابع جهت نمایش یک فایل استفاده می‌شود. در اینجا نیز نام فایل ها می‌توانند حاوی حرف فاصله نیز باشند. برای مثال:

```
> read test file.txt
hello
2, 3, 5, 7
15
end
```

در این تابع نیز بخشی از نام فایل می‌تواند متغیر باشد.

دستور run

نمره: ۱۰

این دستور برای باز کردن یک اسکریپت به زبان ۴۰۰۱ است.

```
> run <file>
```

برنامه شما باید قادر باشد فایلی که در پارامتر دریافت می‌شود را اجرا کند. دستورات در این فایل به صورت خط به خط نوشته شده‌اند.

مانند توابع قبلی بخشی از نام فایل اسکریپت نیز می‌تواند متغیر باشد.

مثالی از یک اسکریپت و اجرای آن (بدون متغیر):

```
01 echo hello this is test script
02 calc - # ^ 5 2 11 3
03 prims 7
04 open output.txt
05 echo output file begins
06 calc + 2 3
07 close
08 echo file closed
09 pause
```

```
01 > run test.script
02 hello this is test script
03 0
04 2 3 5 7
05 File "output.txt" opened
06 output file begins
07 5
08 file "output.txt" closed
09 file closed
10 Please press ENTER to continue...
```

موفق باشید