بسمه‌تعالی

ساختمان‌داده‌ها

نیمسال دوم ۹۷-۹۶

مدرس: دکتر امیر جهانگرد رفسنجانی

تمرین عملی اول (۱۰۰ نمره) مهلت تحویل: ۱۳۹۷/۱/۲۱

تمرین عملی اول شامل دو قسمت است:

۱. پیاده‌سازی پشته (۳۰ نمره)
۲. تبدیل عبارت میانوندی به پسوندی با استفاده از پشته (۷۰ نمره)

در ادامه توضیحاتی در رابطه با هر قسمت به همراه نکاتی که باید در انجام تمرین مورد توجه قرار بگیرند، آورده شده است.

۱. پیاده‌سازی پشته

کلاس Stack در فایل stack.py شامل توابع زیر است:

- تابع __init__ بعنوان سازندهٔ[۱] کلاس، یک پشته خالی را ایجاد می‌کند.
- تابع count، تعداد آیتم‌های درون پشته را برمی‌گرداند.
- تابع is_empty، اگر پشته خالی باشد مقدار منطقی True را برمی‌گرداند در غیر اینصورت مقدار منطقی False را برمی‌گرداند.
- تابع push، یک آیتم را در بالای پشته قرار می‌دهد.
- تابع top، اگر پشته خالی باشد آنگاه یک استثناء از کلاس Exception را برمی‌گرداند در غیر اینصورت مقدار آیتم بالای پشته را (بدون حذف آن) نشان می‌دهد.
- تابع pop، اگر پشته خالی باشد آنگاه یک استثناء از کلاس Exception را برمی‌گرداند در غیر اینصورت آیتم بالای پشته را حذف می‌کند و مقدار آن را برمی‌گرداند.
- تابع clear، تمامی آیتم‌های درون پشته را حذف می‌کند بگونه‌ای که یک پشته خالی بوجود می‌آید.

این کلاس را بگونه‌ای تکمیل کنید که هر تابع به درستی وظیفهٔ مربوط به خود را انجام دهد. برای این منظور از نوع داده‌ای[۲] لیست استفاده کنید. جدول ۱ یک نمونه ورودی و خروجی به کلاس Stack را نشان می‌دهد. ورودی شامل تعدادی دلخواه عملیات push و pop است که به دستور end ختم می‌شوند. با دستور push مقداری در پشته قرار می‌گیرد و با دستور pop مقدار بالای پشته حذف و در خروجی چاپ می‌شود. با دستور end نیز تعداد عناصر درون پشته در خروجی چاپ می‌شود سپس تمامی عناصر درون پشته حذف می‌شوند و با یک فاصله از یکدیگر به ترتیب از چپ به راست در خروجی چاپ می‌شوند. در انتها نیز یک حذف از پشته صورت می‌گیرد که بخاطر خالی بودن پشته و ایجاد استثناء، پیام .Stack is empty در خروجی چاپ می‌شود.

[۱] constructor

[۲] data type

جدول ۱:یک نمونه ورودی و خروجی به کلاس Stack

ورودی	خروجی
push 4	۵
push 5	۲
pop	۳ ۴
push 3	Stack is empty.
end	

۲. تبدیل عبارت میانوندی به پسوندی با استفاده از پشته

تابع infix_to_postfix در فایل infix2postfix.py را بگونه‌ای تکمیل کنید که با دریافت یک عبارت میانوندی بعنوان ورودی، شکل پسوندی آنرا برگرداند. ورودی این تابع یک رشته است که عملوندهای آن حروف بزرگ انگلیسی یعنی ABCDEFGHIJKLMNOPQRSTUVWXYZ هستند. عملگرهای آن شامل * (ضرب)، / (تقسیم)، + (جمع) و – (تفریق) هستند. همچنین برای اولویت‌بندی عملگرها بگونه‌ای غیر از اولویت معمول بین آنها از () یعنی پرانتز باز و بسته استفاده می‌شود. جدول ۲ دو نمونه ورودی و خروجی به این تابع را نشان می‌دهد. روند کار بدین صورت است که رشته ورودی به این تابع داده می‌شود و شکل پسوندی آن در خروجی چاپ می‌شود. دقت کنید که بین کاراکترهای رشته ورودی و خروجی هیچ فاصله‌ای وجود ندارد حتماً این موضوع را هنگام پیاده‌سازی رعایت کنید.

جدول ۲: دو نمونه ورودی و خروجی به تابع infix_to_postfix

ورودی	خروجی
(A+B)*C-(D-E)*(F+G)	AB+C*DE-FG+*-
(A+B*C)-D-E*F+G	ABC*+D-EF*-G+

برای انجام تمرین علاوه بر توضیحات داده شده به نکات زیر نیز توجه کنید:

- در هر دو فایل قسمت‌هایی که باید تکمیل شوند، مشخص شده‌اند. در این قسمت‌ها ابتدا دستور pass را حذف کنید سپس شروع به نوشتن دستورات خود کنید.
- برای استفاده از کلاس Stack در فایل infix2postfix.py از دستور import استفاده نکنید زیرا فایل‌ها بصورت جداگانه مورد ارزیابی قرار می‌گیرند بنابراین همانگونه که در این فایل مشخص شده است کلاس Stack را مانند فایل stack.py تکمیل کنید.
- در انتهای هر دو فایل قسمتی با عنوان ”__main__“ == __name__ if وجود دارد. این قسمت برای ارزیابی تمرین شما است که چگونگی آن توضیح داده شده است. اکیداً توصیه می‌شود که پیش از ارسال تمرین از این قسمت برای اطمینان از درستی برنامه خود استفاده کنید.
- توصیه می‌شود که غیر از قسمت‌های تعیین شده، تغییراتی را در فایل‌ها ایجاد نکنید. زیرا ممکن است که ارزیابی تمرین شما را با مشکل مواجه کند.

موفق باشید.