دانشکده مهندسی کامپیوتر

برنامه‌سازی پیشرفته
تمرین های سری هفتم

مدرس: سید صالح اعتمادی
طرح تمرین: امید میرزاجانی

مهلت ارسال:
شنبه ۲۰ اردیبهشت ۹۹

فهرست مطالب

۱ مقدمه

۱.۱ موارد مورد توجه

- توجه داشته باشید که برای کسب نمره‌ی قبولی درس کسب حداقل نصف نمره‌ی هر سری تمرین الزامی می‌باشد.
- مهلت ارسال پاسخ تمرین تا ساعت ۲۳:۵۹ روز اعلام‌شده است. توصیه می‌شود نوشتن تمرین را به روزهای نهایی موکول نکنید.
- همکاری و هم‌فکری شما در حل تمرین مانعی ندارد، اما پاسخ ارسالی هر کس حتما باید توسط خود او نوشته شده باشد.
- مبنای درس، اعتماد بر پاسخ ارسالی از سوی شماست؛ بنابراین ارسال پاسخ در ریپازیتوری گیت شما به این معناست که پاسخ آن تمرین، توسط شما نوشته شده است. در صورت تقلب یا اثبات عدم نوشتار پاسخ حتی یک سوال از تمرین، برای هر دو طرف تقلب‌گیرنده و تقلب‌دهنده نمره‌ی **مردود** برای درس در نظر گرفته خواهد شد.
- توجه داشته باشید که پاسخ‌ها و کدهای مربوط به هر مرحله را بایستی تا قبل از پایان زمان مربوط به آن مرحله، در سایت Azure DevOps (طبق توضیحات کارگاه‌ها و کلاس‌ها) بفرستید. درست کردن `Pull request` و `Complete` کردن `Pull request` و انتقال به شاخه‌ی `master` پس از تکمیل تمرین فراموش نشود!
- پس از پایان مهلت ارسال تا ۲ روز به ازای هر روز تاخیر ۱۰ درصد از نمره مربوط به تمرین کسر خواهد شد و پس از ۲ روز نمره‌ای به تمرین تعلق نخواهد گرفت.
- بعضی از قسمت های تمرین نیاز به پیاده سازی بر روی هر چهار زبان `C#` ، `Python` ، `C++` و `Java` را دارند بعضی هم خیر. بنابراین روبروی هر سوال زبان های مورد نیاز برای پیاده سازی مشخص شده است.

۲ آماده سازی های اولیه

۱.۲ ساخت پروژه ی C#

برای ایجاد پروژه C# کافی است کد زیر را در ترمینال خود اجرا کنید:

```
mkdir A7_cs
cd A7_cs
dotnet new sln
mkdir A7_cs
cd A7_cs
dotnet new console
cd ..
dotnet sln add A7_cs\A7_cs.csproj
mkdir A7_cs.Tests
cd A7_cs.Tests
dotnet new mstest
dotnet add reference ..\A7_cs\A7_cs.csproj
cd ..
dotnet sln add A7_cs.Tests\A7_cs.Tests.csproj
```

۲.۲ قواعد نام گذاری

قواعد نام‌گذاری تمرین را از جدول ۱ مطالعه کنید.

`*` در کل یک دیرکتوری داخل Assignments به نام A۷ بسازید و داخل آن،یک دیرکتوری به نام A۷_cs داشته باشید و فایل های مربوطه را داخل دیرکتوری مربوطه بگذارید.

جدول ۱: قراردادهای نام‌گذاری تمرین

Naming conventions		
Branch	Directory	Pull Request
`fb_A7`	`A7`	`A7`

۳ انواع داده ای

دیمو [۱] برای انجام تمرین های درس گسسته، به چند نوع داده ای جدید نیاز دارد که بتواند خواسته های او را برآورده کند. او تا به حال با آرایه و لیست آشنا شده است، اما الآن با توجه به مسأله، نیازی ندارد که همه نمایه [۲] ها را داشته باشد. گاهی نیاز دارد فقط اول یا آخر داده ها را بدانیم. همانطور که احتمالاً متوجه شدید، در این سری تمرین قرار است انواع داده ای Stack و Queue را پیاده سازی کنیم.

۱.۳ Stack

استک یا پشته، نوعی داده است که در آن دو عمل push ، pop تعریف میشود. اگر بخواهیم مثالی از اطرافمان بزنیم،وقتی چندین کتاب را روی هم میگذاریم، یک استک ساخته ایم. برای مثال اگر بخواهیم کتاب زرد زنگ را برداریم، ابتدا باید کتاب های سبز و آبی به ترتیب برداشته شوند .

در نوع داده ای استک نیز همینگونه است و عمل push به معنای آن است که یک داده در بالای استک قرار داده شود. و عمل pop نیز به معنای برداشتن و حذف بالاترین داده از استک است.

۱.۱.۳ Constructor

کلاس جنریک [۳] MyStack را به گونه ای بنویسید، که دو ویژگی

- Size از نوع `int`
- Values از نوع لیستی از همان نوع داده ای که قرار است، تشکیل شود.

پس از پیاده سازی این کلاس و سازنده اش، تست `StackConstructorTest` پاس خواهد شد.

۲.۱.۳ Push

این متد را به گونه ای پیاده سازی کنید که یک ورودی از نوع داده ای مناسب بگیرد، و اولاً Size را یک واحد افزایش دهد، دوماً این نوع داده ای را به لیست Values اضافه کند.

۳.۱.۳ Pop

این متد را به گونه ای پیاده سازی کنید که اولاً Size را یک واحد کاهش دهد، دوماً آخرین داده ای که اضافه شده است را حذف کند. دقت کنید که پس از انجام این دستور، آن داده باید از Values نیز حذف شود.
پس از پیاده سازی صحیح تست `StackPushPopTest` پاس خواهد شد.

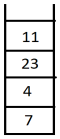
---

[۱] Dimo
[۲] index
[۳] Generic

Print ۴.۱.۳

این متد را بگونه ای پیاده سازی کنید، که داده های استک را در قالب یک رشته برگرداند که ترتیب شان، همان ترتیب خروجشان از استک است . همچنین دقت کنید که در این متد شما مجاز به استفاده از مقادیر Values نیستید و برای به دست آوردن مقادیر ذخیره شده، باید از متد های Push،Pop استفاده کنید. پس از پیاده سازی صحیح تست `StackPrintTest` پاس خواهد شد.

برای مثال خروجی برای استک زیر باید به صورت رشته ی `11 23 4 7` باشد.



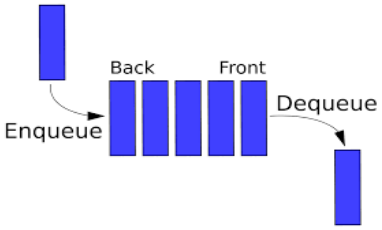
IEnumerable ۵.۱.۳

این کلاس را به گونه ای پیاده سازی کنید که بتوانیم با حلقه for بر روی آن پیمایش کنیم و مقادیر را به همان ترتیب خروجشان از استک برگرداند.

پس از پیاده سازی صحیح تست `StackEnumeratorTest` پاس خواهد شد.

Queue ۲.۳

این نوع داده ای به صف مانند است و بر روی آن دو عمل Dequeue ، Enqueue تعریف میشود. به این صورت که ،Dequeue اولین عضو را خارج میکند و ،Enqueue یک عضو جدید را به انتهای صف اضافه میکند.



Constructor ۱.۲.۳

کلاس جنریک MyQueue را به گونه ای بنویسید، که دو ویژگی

- Size از نوع `int`
- Values از نوع لیستی از همان نوع داده ای که قرار است، تشکیل شود.

پس از پیاده سازی این کلاس و سازنده اش، تست `QueueConstructorTest` پاس خواهد شد.

Enqueue ۲.۲.۳

این متد را به گونه ای پیاده سازی کنید که یک ورودی از نوع داده ای مناسب بگیرد، و اولاً Size را یک واحد افزایش دهد، دوماً این نوع داده ای را به لیست Values اضافه کند.

Pop ۳.۲.۳

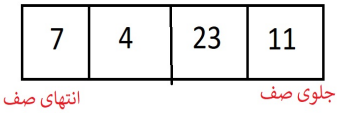
این متد را به گونه ای پیاده سازی کنید که اولاً Size را یک واحد کاهش دهد، دوماً اولین داده ای که اضافه شده است را حذف کند. دقت کنید که پس از انجام این دستور، آن داده باید از Values نیز حذف شود.

پس از پیاده سازی این صحیح تست `EnqueueDequeueTest` پاس خواهد شد.

Print ۴.۲.۳

این متد را بگونه ای پیاده سازی کنید، که داده های صف را در قالب یک رشته برگرداند که ترتیب شان، همان ترتیب خروجشان از صف است . همچنین دقت کنید که در این متد شما مجاز به استفاده از مقادیر Values نیستید و برای به دست آوردن مقادیر ذخیره شده، باید از متد های Enqueue،Dequeue استفاده کنید. پس از پیاده سازی صحیح تست `QueuePrintTest` پاس خواهد شد.

برای مثال خروجی برای استک زیر باید به صورت رشته ی `11 23 4 7` باشد.



IEnumerable ۵.۲.۳

این کلاس را به گونه ای پیاده سازی کنید که بتوانیم با حلقه for بر روی آن پیمایش کنیم و مقادیر را به همان ترتیب خروجشان از صف برگرداند.
پس از پیاده سازی صحیح تست `QueueEnumerableTest` پاس خواهد شد.

۳.۳ تبدیلات

حال که این دو بخش را به طور کامل پیاده سازی کردید، ارتباط بین این دو را نیز مشخص کنید.

IConvertable ۱.۳.۳

این اینترفیس جنریک را به عنوان راهنمایی برای شما پیاده سازی شده است. هدف ، این است که اگر کلاسی این اینترفیس را داشت، قابل تبدیل به شی ای از نوع T نیز باشد.

```
public interface IConvertable<T>
{
    T Convert();
}
```

این اینترفیس را به ویژگی های دو کلاس MyStack ، MyQueue اضافه کنید و آن را برای هر کلاس به گونه ای پیاده سازی کنید که بتواند این دو نوع داده ای را به یکدیگر تبدیل کند. پس از پیاده سازی صحیح، تست های `StackConvertTest` و `QueueConvertTest` پاس خواهد شد. برای روشن شدن مسأله به تست ها رجوع کنید.

Operator + ۲.۳.۳

عملگر + را برای کلاس MyStack به گونه ای پیاده سازی کنید، که قابلیت جمع زدن با MyStack ، MyQueue را داشته باشد. جمع زدن برای این کلاس هم به این صورت تعریف میشود که تمامی اعضای متغیر دوم را، به مجموعه اعضای خود ، به همان ترتیب خارج شدنشان، اضافه کند پس از پیاده سازی صحیح، تست `StackPlusTest` پاس خواهد شد.
به طور مشابه این عملگر + را برای کلاس MyQueue به صورتی پیاده سازی کنید که تست `QueuePlusTest` پاس خواهد شد.
روشن شدن این مسأله به خواندن تست ها وابسته است.

سرزنده و متفکر باشید.