دانشکده‌ی مهندسی کامپیوتر

مبانی برنامه‌سازی کامپیوتر
امتحان عملی ۱

سید صالح اعتمادی *

وقت امتحان: ۳ ساعت

*تشکر ویژه از آقای علی حیدری که نسخه اولیه این قالب را در ترم دوم سال تحصیلی ۹۷-۹۸ برای درس برنامه‌سازی پیشرفته تهیه کردند.

فهرست مطالب

۱ آماده‌سازی ۳
۱.۱ نکات مورد توجه ۳
۲.۱ آماده‌سازی‌های اولیه ۳
۱.۲.۱ آماده‌سازی‌های مربوط به git ۳
۲.۲.۱ آماده‌سازی‌های مربوط به VSCode ۴

۲ پیاده‌سازی ۴
۱.۲ آماده سازی تست ۴
۱.۱.۲ فعال‌سازی تست‌ها ۴
۲.۱.۲ فرستادن تست در `Azure DevOps` ۴
۲.۲ پیاده‌سازی توابع ۴
۱.۲.۲ تست `test_q0_pythagoras` ۵
۲.۲.۲ تست `test1_is_pythagoras` ۵
۳.۲.۲ تست `test_q2_get_last_index` ۵
۴.۲.۲ تست `test_q3_dedup` ۵
۵.۲.۲ تست `test_q4_sum_of_dividable_indexes` ۵
۶.۲.۲ تست `test_q5_concat_of_prime_positions` ۵
۷.۲.۲ تست `test_q6_contains_pattern` ۵
۸.۲.۲ تست `test_q7_reverse_dict_mapping` ۵
۹.۲.۲ تست `test_q8_get_odd_file_lines` ۵

۳ ارسال ۶
۱.۳ ساخت Pull Request ۶

۱ آماده‌سازی

۱.۱ نکات مورد توجه

- صدا و صفحه نمایش شما باید از طریق نرم‌افزار Flashback recorder به طور کامل از ابتدا تا انتهای امتحان ضبط و ذخیره شود. دقت کنید که پس از نصب نرم‌افزار، در قسمت تنظیمات کیفیت ضبط را ۱ فریم بر ثانیه قرار دهید. ویدیوی امتحان بعد از امتحان جمع‌آوری خواهد شد.
- لازم است از ابتدا تا انتهای امتحان در جلسه Teams با وب‌کم روشن حضور داشته باشید.
- دیدن هرگونه کد از روی اینترنت یا غیراینترنت مجاز نیست. پاسخ ارسالی هر کس حتما باید توسط خود او و بدون دیدن هیچ کد دیگری نوشته شده باشد. کمک گرفتن از دیگران در طول مدت امتحان مجاز نیست و منجر به درج نمره‌ی **مردود** برای این درس می‌شود.
- تنها منبع مجاز برای امتحان استفاده از فایل note.txt است که از قبلا آماده کرده باشید و همراه با امتحان ارسال نمایید.
- ارتباط با هر فردی بصورت حضوری یا مجازی تقلب محسوب می‌شود. تنها ارتباط مجاز از طریق نرم‌افزار Teams و از روش فوق است.
- معیار ارزیابی امتحان فقط کدی است که در AzureDevOps با روشی که در ادامه آمده بارگزاری شده است.
- خروج از جلسه Teams یا محدوده تصویر وب‌کم در طول امتحان مجاز نیست.
- خوردن و آشامیدن بلا مانع است.

۲.۱ آماده‌سازی‌های اولیه

قواعد نام‌گذاری آزمون را از جدول ۱ مطالعه کنید.

جدول ۱: قراردادهای نام‌گذاری آزمون

Naming conventions			
Feature Branch	Directory	Pull Request Title	Target Branch
`fb_E2`	E2	E2	`holymaster`

۱.۲.۱ آماده‌سازی‌های مربوط به git

اگر سر کلاس و کارگاه چند بار مفاهیم و روش کار با git آموزش داده شد اما بار دیگر در اینجا کارهایی را که باید در ابتدای آزمون انجام دهید را مرور می‌کنیم.

✓ ابتدا به شاخه‌ی `master` بروید و از یکسان بودن این شاخه با سرور اطمینان حاصل کنید.

```
C:\git\FC00011>git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.

C:\git\FC00011>git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

C:\git\FC00011>git pull
Already up to date.

C:\git\FC00011>
```

✓ سپس این کار را برای شاخه `holymaster` تکرار کنید.

```
C:\git\FC00011>git checkout holymaster
Switched to branch 'holymaster'
Your branch is up to date with 'origin/holymaster'.

```

```
C:\git\FC00011>git status
On branch holymaster
Your branch is up to date with 'origin/holymaster'.

nothing to commit, working tree clean

C:\git\FC00011>git pull
Already up to date.

C:\git\FC00011>
```

✓ یک شاخه‌ی جدید با نام `fb_E2` بسازید و تغییر شاخه دهید.

```
C:\git\FC00011>git branch fb_E2

C:\git\FC00011>git checkout fb_E2
Switched to branch 'fb_E2'

C:\git\FC00011>git status
On branch fb_E2
nothing to commit, working tree clean

C:\git\FC00011>
```

توصیه می‌شود پس از پیاده‌سازی هر تست تغییرات انجام شده را `commit` و `push` کنید.

۲.۲.۱ آماده‌سازی‌های مربوط به VSCode

پوشه‌ای با نام `E2` در ریشه گیت درست کرده و فایل تست `exam2_test.py` را در آن قرار دهید. سپس پوشه `E2` را با VSCode باز کنید.

۲ پیاده‌سازی

۱.۲ آماده سازی تست

سوال‌های امتحان بصورت تعدادی تست طراحی شده‌اند که لازم است تابع لازم برای پاس شدن تست را پیاده‌سازی کنید. همه تست‌ها comment شده و pytest برای رد کردن و عدم اجرای تست تنظیم شده.

۱.۱.۲ فعال‌سازی تست‌ها

در قدم اول کامنت‌های مربوط به تست‌ها را یکی‌ـیکی برداشته و از از شناخته شدن تست توسط pytest در VSCode اطمینان حاصل کنید. برای این منظور ابتدا کامنت‌های مربوط به یک تست را بردارید. سپس تست را مطالعه کنید. نام تابع مورد تست و پارامترهای ورودی و نوع مقدار برگشتی تابع مورد تست را تشخیص دهید. سپس در فایل `exam2.py` تابع را پیاده‌سازی کنید. اگر این اولین تست است توسط دستور `Python: Configure Tests` در VSCode بستر تست pytest را فعال کنید. سپس از شناخته شدن تستی که کامنت آن را برداشتید اطمینان حاصل کنید. این فرایند را تا براشته شدن کلیه کامنت‌های تست ادامه دهید.

۲.۱.۲ فرستادن تست در `Azure DevOps`

پس از شناخته شدن کلیه تست‌ها در VSCode کد خود را add/commit/push کرده و سپس در `Azure DevOps` یک `Pull Request` برای بردن این تغییرات از شاخه `fb_E2` به شاخه `holymaster` درست کنید. چنانچه شاخه `holymaster` و `branch policy` مربوط به آنرا بدرستی تنظیم کرده باشید، بیلد مرتبط با این `Pull Request` باید موفقیت آمیز باشد.

۲.۲ پیاده‌سازی توابع

از تست شماره یک شروع کرده و دستور `@pytest.mark.skip` قبل از تست را برداشته تا تست فعال شود. سپس تست را اجرا کرده و از عدم اجرای موفقیت‌آمیز آن اطمینان حاصل کنید. سپس تابع مورد استفاده تست را بگونه‌ای پیاده‌سازی کنید که تست با موفقیت پاس شود.

۱.۲.۲ تست `test_q0_pythagoras`

تابع `q0_pythagoras`
طول دو ضلع مجاور زاویه قائمه در یک مثلث قائم‌الزاویه را به عنوان ورودی دریافت کرده و با استفاده از رابطه فیثاغورث طول وتر را برمی‌گرداند.

۲.۲.۲ تست `test_q1_is_pythagoras`

تابع `q1_is_pythagoras` سه عدد a, b, c را به عنوان پارامتر دریافت کرده و درصورتیکه رابطه فیثاغورث بین این ۳ عدد برقرار باشد True و در غیر اینصورت False برمی‌گرداند.

۳.۲.۲ تست `test_q2_get_last_index`

تابع `q2_get_last_index` یک لیست و یک عدد به عنوان پارامتر ورودی دریافت کرده و اندیس محل عدد در لیست را برمی‌گرداند. چنانچه عدد بیش از یک بار تکرار شده باشد، بزرگترین اندیس را برمی‌گرداند. چنانچه عدد پیدا نشود، عدد منفی یک را به عنوان اندیس برمی‌گرداند.

۴.۲.۲ تست `test_q3_dedup`

تابع `q3_dedup` یک لیست به عنوان ورودی دریافت کرده و عناصر منحصر به فرد آنرا به ترتیبی که در لیست آمده‌اند برمی‌گرداند.

۵.۲.۲ تست `test_q4_sum_of_dividable_indexes`

تابع `q4_sum_of_dividable_indexes` یک لیست و یک عدد از ورودی دریافت کرده و مجموع اعدادی که اندیس آنها بر عدد ورودی بخش‌پذیر است را برمی‌گرداند.

۶.۲.۲ تست `test_q5_concat_of_prime_positions`

تابع `q5_concat_of_prime_positions` یک رشته حرفی از ورودی دریافت کرده و رشته حرفی شامل حروفی که در اندیس‌های «اول» قرار دارند را برمی‌گرداند. مثلا اندیس‌های ۲، ۳، ۵، ... .

۷.۲.۲ تست `test_q6_contains_pattern`

تابع `q6_contains_pattern` دو لیست به عنوان پارامتر دریافت می‌کند. چنانچه لیست دوم در لیست اول موجود باشد، True و در غیر اینصورت False برمی‌گرداند. وجود لیست دوم در لیست اول به این معنی است که تمام عناصر لیست دوم، بدون فاصله و پشت سر هم در لیست اول آمده باشند. برای این سوال نباید از هیچ‌گونه توابع آماده استفاده کنید. چنانچه نیاز به تابع اضافه دارید، خودتان می‌توانید پیاده‌سازی کنید.
پیاده‌سازی این تابع نیاز به دقت دارد. اگر گیر کردید، اول سوال‌های بعدی را حل کنید بعد برگردید سر این سوال.

۸.۲.۲ تست `test_q7_reverse_dict_mapping`

تابع `q7_reverse_dict_mapping` یک دیکشنری به عنوان ورودی دریافت کرده و یک دیکشنری جدید برگرداند که جای کلید و مقدار در آن عوض شده باشد.

۹.۲.۲ تست `test_q8_get_odd_file_lines`

تابع `q8_get_odd_file_lines` نام یک فایل متنی را به عنوان پارامتر ورودی دریافت کرده و یک فایل جدید با نام دلخواه تولید کرده که شامل خطوط فرد فایل ورودی می‌باشد. شماره خطوط از یک شروع می‌شود. یعنی مثلا محتوای خطوط شماره ۱، ۳، ۵، ... در فایل خروجی موجود باشد. دقت کنید که «نام فایل خروجی» خواسته شده. برای راحتی کار و عدم برخورد به اشکال نام پوشه به نام فایل اضافه نکنید. فقط نام فایل را برگردانید.

۳ ارسال

اگر موفق به پاس شدن تستی نشدید دستور مربوط به عدم اجرای تست را قبل از تست باقی بگذارید. پس از پیاده‌سازی توابع و پاس شدن تست‌هایی که فرصت کردین، نوبت به ارسال آنها میرسد. مثل قبل تغییرات را در شاخه `fb_E2` add/commit/push کنید.

۱.۳ ساخت Pull Request

با مراجعه به سایت Azure DevOps لز موفقیت بیلد برای Pull Request که در مرحله اول درست کردید اطمینان حاصل کنید و آنرا کامل کنید. دقت کنید که گزینه‌ی Delete source branch **نباید** انتخاب شود.