# برنامه‌سازی پیشرفته

**تمرین کامپیوتری شماره ۴**

مدرس: رامتین خسروی

طراحان: کوروش علی‌نقی، مجید صادقی نژاد، فرجاد فلاح، مهراد لیویان، عرفان فلاحتی، پارسا سعیدنیا، پریا پاسه‌ورز، مهدی نائینی، ریحانه عین اللهی

مهلت تحویل: شنبه ۶ اردیبهشت ۱۴۰۴، ساعت ۲۳:۵۹

## مقدمه

هدف از این تمرین آشنایی شما با مفاهیم اولیه **طراحی شیءگرا**[1] و طراحی یک نرم‌افزار به کمک آن است. از آنجایی که استفاده و درک درست این مفاهیم در پیاده‌سازی سایر تمرین‌های این درس لازم است، پیشنهاد می‌شود به این تمرین زمان کافی را اختصاص دهید.

استفاده از نرم‌افزاری که در این تکلیف می‌نویسید از طریق وارد کردن تعدادی دستور ورودی و دریافت خروجی صورت می‌گیرد. **توصیه می‌شود نوشتن برنامه را به صورت دستور به دستور جلو ببرید.** یک دستور را کامل کنید، درستی کد خود را بیازمایید و بعد به پیاده‌سازی دستور بعد بپردازید. به این ترتیب حتی اگر در مهلت تعیین شده موفق به تکمیل تمام تکلیف نشدید، بخشی از کارکردها را کامل کرده‌اید. با این هدف، ترتیب مناسبی از دستورها را برای پیاده‌سازی تعریف کنید. به عبارت دیگر، اولین دستور که در اینجا توضیح داده شده لزوماً مناسب‌ترین دستور برای شروع پیاده‌سازی نیست.

[1] Object-Oriented Design

# قلم‌یوتی

## شرح مسئله

لطفعلی که از پشت کنکوری بودن خسته شده و رویای تحصیل در دانشگاه تهران در سر دارد، قصد دارد این بار جدی باشد و حسابی برای کنکور درس بخواند! وظیفه شما این است که به لطفعلی کمک کنید تا بتواند برای خودش آزمون‌های آزمایشی بسازد، در آن‌ها شرکت کند و نتیجه آزمون خود را ببیند.

در این تمرین، شما یک بانک سوال دارید که دارای سوالاتی با درجات سختی متفاوت است. به لطفعلی کمک کنید تا بتواند آزمون شخصی‌سازی‌شده خودش را بسازد و نتیجه آن را تحلیل کند.

## بانک سوال

همانطور که گفته شد، آزمون‌ها باید از بانک سوال تهیه شوند. سوالاتِ بانک سوال، به همراه درجه سختی و پاسخشان در فایل CSV قرار خواهند گرفت. تعدادی فایل CSV برای نمونه در صفحه درس برای شما قرار داده شده است.

توجه کنید که شماره گزینه‌ها همواره از ۱ شروع می‌شوند و نه ۰.

تضمین می‌شود میزان `difficulty` یکی از ۳ مقدار `easy`، `medium` و `hard` است.

نمونه محتوای فایل CSV

```
question_text,option1,option2,option3,option4,correct_answer,difficulty,subject
What is the capital of Norway?,Copenhagen,Oslo,Tabriz,Helsinki,2,easy,Geography
2 + 2 = ?,4,5,196883,934782,1,hard,Maths
```

مسیر فایل CSV در زمان اجرا به برنامه شما به عنوان ورودی خط فرمان[2] داده می‌شود.

[2] Command-line argument

# دستورات

## ساخت قالب آزمون

کاربر می‌تواند قالب آزمونی شخصی‌سازی‌شده برای خودش بسازد. در ادامه او می‌تواند از روی این قالب‌ها آزمون مورد نظرش را تولید و در آن‌ها شرکت کند. دستور ایجاد یک قالب جدید، create_template است؛ کاربر می‌تواند با ورودی‌های مختلفی که به این دستور می‌دهد، تعداد، سطح سوالات و مباحث آن را تعیین کند و قالب خود را شخصی‌سازی کند. نام هر قالب حتما باید یکتا باشد و در ادامه نیز هر قالب با نام آن شناخته خواهد شد.

تضمین می‌شود که موضوع درس، سطح سوال و تعداد سوالات به درستی وارد می‌شوند (تعداد سوالات همواره عددی مثبت است).

نام قالب بین دو کاراکتر سینگل‌کوتیشن (') قرار می‌گیرد و می‌تواند حاوی کاراکتر فاصله نیز باشد.

| قالب دستور |
| --- |
| `create_template ‘<template_name>’ subject1:difficulty:count subject2:difficulty:count ...` |
| قالب خروجی |
| `Template ‘<template_name>’ was created successfully. \| Duplicate name: ‘<template_name>’` |

| نمونه ورودی معتبر |
| --- |
| `create_template ‘Saturday mornings’ Maths:easy:2 Maths:hard:1 Physics:hard:3` |
| نمونه خروجی معتبر |
| `Template ‘Saturday mornings’ was created successfully.` |

در صورتی که نام قالب وارد شده تکراری بود، کاربر با خطا مواجه می‌شود:

| نمونه ورودی نامعتبر |
| --- |
| `create_template ‘Duplicate template name’ Maths:easy:10` |
| نمونه خروجی نامعتبر |
| `Duplicate name: ‘Duplicate template name’` |

## ساخت آزمون

برای استفاده مفید از بانک سوال، کاربر می‌تواند از روی قالبی که قبلاً تعریف کرده است، برای خود آزمونی شخصی‌سازی‌شده با استفاده از سوالات موجود در بانک سوال بسازد. دقت کنید که کاربر می‌تواند از یک قالب، چندین آزمون مختلف بسازد. دستور انجام این کار، `generate_test` است؛ این دستور با گرفتن قالب مورد نظر کاربر و نام آزمون جدید، آزمون خواسته شده را تولید می‌کند. نام هر آزمون نیز همانند نام قالب‌ها باید یکتا باشد. تضمین می‌شود نام آزمون وارد شده، تکراری نیست. همچنین، تضمین می‌شود در بانک سوال، به تعداد کافی سوال برای ساخت آزمون با قالب داده‌شده وجود دارد.

نکته مهم هنگام ساختن آزمون، روش انتخاب سوال‌ها از بانک سوال است. در بانک سوال تعداد زیادی سوال به ازای هر درس و سطح وجود دارد. هدف ما اولویت دادن به سوالاتی است که کاربر در گذشته نتوانسته است به آن‌ها پاسخ درست دهد. به این منظور بر اساس تعداد دفعاتی که کاربر سوالی را نزده رها کرده است، تعداد پاسخ‌های درست و تعداد پاسخ‌های غلط (برای آن سوال)، اولویتی برای هر سوال در نظر می‌گیریم. هنگام ساختن آزمون، سوالات با اولویت بالاتر را انتخاب می‌کنیم:

$$priority = 3 \times numOfIncorrects + 1 \times numOfBlanks - 2 \times numOfCorrects$$

در ابتدا تعداد نادرست‌ها، نزده‌ها و درست‌ها همگی برابر صفر است. بنابراین اولویت همه سوالات در آغاز برابر صفر خواهد بود. برای انتخاب میان سوالاتی که اولویت یکسان دارند، سوالاتی که متنشان کوچک‌تر از بقیه هستند انتخاب می‌شوند. توجه کنید که در تمام صورت پروژه، تمام مقایسه‌های بین دو رشته همانند توضیحات پروژه دوم، بر مبنای کد ASCII هستند.

نام آزمون و قالب بین دو کاراکتر سینگل‌کوتیشن (') قرار می‌گیرند و می‌توانند حاوی کاراکتر فاصله نیز باشند.

| قالب دستور |
| --- |
| `generate_test ‘<test_name>’ ‘<template_name>’` |
| قالب خروجی |

```
Test ‘<test_name>’ was generated successfully. | Could not find
template: ‘<template_name>’
```

| نمونه ورودی معتبر |
| --- |
| `generate_test ‘my first test’ ‘Saturday mornings’` |
| **نمونه خروجی معتبر** |
| `Test ‘my first test’ was generated successfully.` |

در صورتی که قالبی با نام وارد شده وجود نداشته باشد، کاربر با خطا مواجه می‌شود:

| نمونه ورودی نامعتبر دوم |
| --- |
| `generate_test ‘my second test’ ‘Friday mornings’` |
| **نمونه خروجی نامعتبر دوم** |
| `Could not find template: ‘Friday mornings’` |

## شرکت در آزمون ساخته شده

به کمک این دستور، کاربر می‌تواند در آزمونی که پیش‌تر با استفاده از دستور generate_test ساخته بود، شرکت کند و به سوالات پاسخ دهد. تضمین می‌شود کاربر در آزمونی که پیش‌تر در آن شرکت کرده است، شرکت نمی‌کند.

ترتیب نمایش سوالات در آزمون، بر اساس نام موضوع درس (به صورت صعودی) و در بین سوالات با موضوع یکسان، بر اساس متن سوال (به صورت صعودی) می‌باشد. به طور مثال در نمونه بالا، نخست سوالات مربوط به موضوع Geography نمایش داده می‌شوند و سپس سوالات با موضوع Maths.

همچنین در حین آزمون، کاربر می‌تواند با وارد کردن previous، به سوال قبلی برود. در این صورت، باید آخرین پاسخی که کاربر برای آن سوال انتخاب کرده نیز به او نمایش داده شود (به رشته "previous" در مثال دقت کنید).

توجه کنید که گزینه‌های هر سوال خط به خط با **چهار کاراکتر فاصله** در ابتدایشان چاپ می‌شوند. صورت سوال و گزینه‌ها هر کدام بعد از **یک کاراکتر فاصله** بعد از عدد مربوطه چاپ می‌شوند. همچنین قبل از سوال اول و و بعد از تمام سوال‌ها (منظور بعد از پاسخ معتبر دادن و عبور از هر سوال است) یک خط، فاصله وجود دارد.

نام آزمون بین دو کاراکتر سینگل‌کوتیشن (') قرار می‌گیرد و می‌تواند حاوی کاراکتر فاصله نیز باشد.

قالب دستور

```
attend ‘<test_name>’
```

قالب خروجی

```
Could not find test: ‘<test_name>’ |
<test_name>:

1) Question1
    1. option1
    2. option2
    3. option3
    4. option4
Your answer:
2) Question2
    1. option1
    2. option2
    3. option3
    4. option4
Your answer:
```

```
...

n) QuestionN
    1. option1
    2. option2
    3. option3
    4. option4
Your answer:
Finished <test_name>.
```

نمونه ورودی

```
attend ‘my first test’
```

نمونه خروجی

```
my first test:

1) What is the capital of Norway?
    1. Copenhagen
    2. Oslo
    3. Tabriz
    4. Helsinki
Your answer: 2

2) 2 + 2 = ?
    1. 4
    2. 5
    3. 196883
    4. 934782
Your answer: previous

1) What is the capital of Norway?
    1. Copenhagen
    2. Oslo <-
    3. Tabriz
    4. Helsinki
```

```
Your answer: 3

2) 2 + 2 = ?
    1. 4
    2. 5
    3. 196883
    4. 934782
Your answer: Two and two always makes a five
Invalid answer, please try again.
Your answer: 2

Finished my first test.
```

قسمت‌هایی که با رنگ آبی نوشته شده‌اند، توسط کاربر وارد می‌شوند.

توجه کنید که سوال بعدی، تنها در صورتی چاپ می‌شود که کاربر به سوال قبلی پاسخ داده باشد. به طور مثال در نمونه بالا، پس از آن که کاربر یک پاسخ معتبر به سوال ۱ بدهد، سوال ۲ برای او چاپ می‌شود. منظور از پاسخ معتبر، یکی از اعداد ۱ تا ۴، خالی (صرفاً فشردن اینتر بدون تایپ کردن هیچ کاراکتری) و کلمه `previous` است.

وارد کردن `previous` در سوال اول نیز پاسخ نامعتبر است و باید مانند پاسخ‌های نامعتبر مدیریت شود.

خالی گذاشتن پاسخ به منزله رها کردن آن سوال و جواب ندادن به آن است. در صورتی که پاسخ وارد شده معتبر نباشد، پیغام `"Invalid answer, please try again."` چاپ شده و کاربر باید دوباره پاسخ خود را وارد کند.

در صورتی که آزمونی با نام داده شده ساخته نشده باشد، کاربر با خطا مواجه می‌شود:

| نمونه ورودی نامعتبر |
| --- |
| `attend 'Easy OS test'` |
| **نمونه خروجی نامعتبر** |
| `Could not find test: 'Easy OS test'` |

## ساخت خودکار آزمون

قلم‌یوتی یک بانک سوال معمولی نیست! با قلم‌یوتی می‌توانید آزمون‌های ۱۰ سواله متناسب با عملکرد خودتان بسازید! این آزمونک‌ها، همانند آزمون‌های دیگر هستند و با دستور `auto_generate` ساخته می‌شوند. با این

تفاوت که برای ساخت آن‌ها، نیازی به قالب ندارید. پس از ساخت آزمونک (همانند سایر آزمون‌ها)، می‌توان با دستور `attend` در آن‌ها شرکت کرد.

نام آزمون بین دو کاراکتر سینگل‌کوت (') قرار می‌گیرد و می‌تواند حاوی کاراکتر فاصله نیز باشد.

| قالب دستور |
| --- |
| `auto_generate ‘<test_name>’` |
| **قالب خروجی** |
| `Test ‘<test_name>’ was generated successfully.` |

| نمونه ورودی |
| --- |
| `auto_generate ‘Routine quiz’` |
| **نمونه خروجی** |
| `Test ‘Routine quiz’ was generated successfully.` |

الگوریتم ساخت این آزمون، به شرح زیر است:

- ابتدا، دو درس با کمترین میانگین درصد جواب‌های درست را پیدا و انتخاب کنید: تعداد سوالاتی از این مبحث که کاربر به آن‌ها جواب درست داده است، تقسیم بر تعداد سوالاتی از این مبحث که کاربر در آزمون‌های مختلف با آن‌ها روبه‌رو شده است (سوالات تکراری هم شمرده شوند).
- برای درس با کمترین درصد جواب‌های درست، ۶ سوال (۳ سوال راحت، ۲ سوال متوسط و ۱ سوال سخت) و برای درس دوم، ۴ سوال (۲ سوال راحت، ۱ سوال متوسط و ۱ سوال سخت) در نظر بگیرید. بین دروس با درصد برابر، از درس با متن کوچک‌تر (مقایسه ASCII) استفاده کنید.
- حالا با این قالب، آزمونک جدید را (مشابه دستور ساخت آزمون و با همان اولویت‌بندی برای سوالات) ایجاد کنید.

## دستورات گزارش‌گیری

کاربر با دستورات متنوعی، می‌تواند نتیجه آزمون‌های خود را به تفکیک مبحث و میزان سختی ببیند. دقت کنید که کاربر ممکن است به یک سوال در چند آزمون، جواب‌های مختلفی داده باشد. در همه گزارش‌گیری‌ها، تمام این جواب‌ها باید در نظر گرفته شوند. به طور مثال اگر کاربر در یک آزمون به سوال 2 + 2 = ?، جواب ۲ داده بود و در

یک آزمون دیگر، جواب ۳، هر دوی این جواب‌ها در اعداد گزارش شده در این دستورات تاثیر دارند. همچنین، نتایج آزمونک‌های خودکار ساخته شده نیز همانند سایز آزمون‌ها در این گزارش‌گیری‌ها اثر دارند.

مقدار Score در تمام گزارش‌ها، برابر است با درصد تعداد پاسخ‌های درست نسبت به کل پاسخ‌های آن موضوع (یا کل آزمون). همچنین این عدد، باید همواره با دقت ۳ رقم اعشار **قطع** شود. دقت کنید که در محاسبه تمام پاسخ‌ها، باید دفعاتی که کاربر با سوال مواجه شده و به آن پاسخ نداده را نیز در نظر بگیرید.

توجه کنید که قالب خروجی (از جمله املای کلمات و خطوط خالی) باید برای تمام دستورات در پیاده‌سازی شما درست باشد.

## گزارش کلی

با وارد کردن این دستور، گزارشی جامع از تمام آزمون‌هایی که کاربر تا به حال در آن‌ها شرکت کرده، به او نشان داده می‌شود. ترتیب نمایش موضوعات به صورت صعودی (بر اساس نام موضوع) می‌باشد.

قالب دستور

```
report all
```

قالب خروجی

```
Total report:

Subject1: <num_of_corrects> corrects, <num_of_incorrects> incorrects
and <num_of_blanks> blanks. Score: <num_of_corrects/total>.
Subject2: <num_of_corrects> corrects, <num_of_incorrects> incorrects
and <num_of_blanks> blanks. Score: <num_of_corrects/total>.
...
SubjectN: <num_of_corrects> corrects, <num_of_incorrects> incorrects
and <num_of_blanks> blanks. Score: <num_of_corrects/total>.

Total results: <num_of_corrects> corrects, <num_of_incorrects>
incorrects and <num_of_blanks> blanks.
Total score: <num_of_corrects/total>.
```

نمونه ورودی

```
report all
```

نمونه خروجی

```
Total report:

Chemistry: 0 corrects, 987 incorrects and 13 blanks. Score: 0.000%.
Maths: 18 corrects, 1 incorrects and 1 blanks. Score: 90.000%.
Physics: 1 corrects, 999 incorrects and 0 blanks. Score: 0.001%.

Total results: 19 corrects, 1987 incorrects and 14 blanks.
Total score: 0.940%.
```

## گزارش آزمون

کاربر با وارد کردن این دستور، می‌تواند نتیجه آزمون داده شده را ببیند. تضمین می‌شود آزمونی با نام داده شده ساخته شده است و کاربر نیز در آن شرکت کرده است. ترتیب نمایش موضوعات به صورت صعودی می‌باشد.

قالب دستور

```
report test ‘<test name>’
```

قالب خروجی

```
Results for <test name>:

Subject1: <num_of_corrects> corrects, <num_of_incorrects> incorrects
and <num_of_blanks> blanks. Score: <num_of_corrects/total>.
Subject2: <num_of_corrects> corrects, <num_of_incorrects> incorrects
and <num_of_blanks> blanks. Score: <num_of_corrects/total>.
...

Total results: <num_of_corrects> corrects, <num_of_incorrects>
incorrects and <num_of_blanks> blanks.
Total score: <num_of_corrects/total>.
```

نمونه ورودی

```
report test ‘Konkour preparation’
```

نمونه خروجی

```
Results for Konkour preparation:

Maths: 18 corrects, 1 incorrects and 1 blanks. Score: 90.000%.
Physics: 1 corrects, 499 incorrects and 0 blanks. Score: 0.200%.

Total results: 19 corrects, 500 incorrects and 1 blanks.
Total score: 3.653%.
```

### گزارش به تفکیک آزمون

کاربر با وارد کردن این زیردستور می‌تواند تاریخچه آزمون‌هایی که در آن‌ها شرکت کرده است را مشاهده کند. ترتیب چاپ شدن آزمون‌ها در خروجی، به همان ترتیبی است که کاربر در آن‌ها شرکت کرده است (و نه لزوماً ترتیبی که آن‌ها را ساخته است).

قالب دستور

```
report tests
```

قالب خروجی

```
Results per attended tests:

test1_name: <num_of_corrects> corrects, <num_of_incorrects>
incorrects and <num_of_blanks> blanks. Score:
<num_of_corrects/total>.
test2_name: <num_of_corrects> corrects, <num_of_incorrects>
incorrects and <num_of_blanks> blanks. Score:
<num_of_corrects/total>.
...
```

نمونه ورودی

```
report tests
```

نمونه خروجی

```
Results per attended tests:

Take home quiz: 23 corrects, 3 incorrects and 4 blanks. Score: 76.666%.
OS Exam: 0 corrects, 9999 incorrects and 9999 blanks. Score: 0.000%.
```

### گزارش مبحث

از آنجا که لطفعلی باید بداند که در کدام مبحث مشکل دارد، باید بتواند با وارد کردن این دستور برای هر کدام از مباحث تاریخچه آن مبحث را ببیند. تضمین می‌شود مبحثی با نام وارد شده وجود دارد.

قالب دستور

```
report subject <subject_name>
```

قالب خروجی

```
Results for <subject_name>:

Easy: <num_of_corrects> corrects, <num_of_incorrects> incorrects and <num_of_blanks> blanks.
Medium: <num_of_corrects> corrects, <num_of_incorrects> incorrects and <num_of_blanks> blanks.
Hard: <num_of_corrects> corrects, <num_of_incorrects> incorrects and <num_of_blanks> blanks.

Total score: <num_of_corrects/total>.
```

نمونه ورودی

```
report subject Math
```

نمونه خروجی

```
Results for Math:

Easy: 4 corrects, 1 incorrects and 0 blanks.
Medium: 3 corrects, 2 incorrects and 0 blanks.
Hard: 1 corrects, 1 incorrects and 3 blanks.

Total score: 66.666%.
```

# نکات و نحوه تحویل

- تحویل این تمرین در سامانه گیت‌هاب انجام می‌شود. برای انجام تمرین لطفا از طریق این لینک وارد شوید، پس از آن باید شماره دانشجویی خود را انتخاب کنید (دقت کنید که با کمک این شماره دانشجویی به شما نمره خواهیم داد، لطفا در انتخاب درست شماره دانشجویی حتما دقت کنید، در صورتی که به مشکل خوردید با دستیاران در ارتباط باشید). پس از آن به صفحه‌ای منتقل می‌شوید که در آنجا می‌توانید تمرین جدید را قبول کنید، پس از قبول کردن تمرین یک مخزن[3] در AP Assignments برای شما ساخته می‌شود و باید کدهای خود را در آنجا قرار دهید.
- پس از انجام تمرین و بارگذاری در گیت‌هاب، کد Hash آخرین کامیت[4] را به همراه شماره دانشجویی خود در سامانه ایلرن آپلود کنید (در خط اول شماره دانشجویی، پس از آن از Enter استفاده کنید و به خط بعد بروید و پس از آن Hash آخرین کامیت). نمونه متن خواسته شده در سامانه ایلرن (بخش <last_commit_hash> و <sid> را جایگزین کنید):

```
<sid>
<last_commit_hash>
```

نمونه:

```
810100000
bad8fbcdcfa3b9feb371a31e0c370150aa870b18
```

- دقت کنید که عدم رعایت ساختار گفته شده در آپلود یا تغییر ساختار فایل‌ها در مخزن (می‌توانید به دلخواه خود فایل اضافه کنید و ... اما اسم و ساختار فایل‌هایی که در ابتدا به شما داده می‌شود نباید تغییر کند) باعث کسر 5 درصد از نمره شما خواهد شد.
- پروژه شما باید به صورت چند فایلی[5] و با استفاده از makefile پیاده‌سازی شده باشد. هدف اصلی پروژه یادگیری شی‌گرایی بوده و پیاده‌سازی به صورت چند فایلی صرفا برای آشنایی شما با این مفهوم می‌باشد. دقت کنید در پروژه‌های بزرگ‌تر، شما از ابتدا باید فایل‌ها و اجزای مختلف پروژه را تشخیص داده و آن را پیاده‌سازی کنید؛ با این حال برای سادگی بیش‌تر و به منظور کسب تجربه، بهتر است این پروژه را ابتدا به صورت یک فایل پیاده‌سازی کرده و تمرکز خود را روی طراحی شی‌گرا بگذارید؛ پس از تشخیص کلاس‌ها و پیاده‌سازی پروژه، آن را به چند فایل تقسیم کرده و makefile مناسب را بنویسید.

---

[3] Repository

[4] Commit

[5] multi-file

- دقت کنید که فایل makefile باید در صفحه اول مخزن باشد و در پوشه‌ای قرار نداشته باشد و در آن مشخص کنید که از استاندارد c++20 استفاده می‌کنید.
- نام برنامه قابل اجرای شما باید UTGhalam (بدون هیچ پسوندی مانند exe یا out) باشد و پس از ساخته شدن در کنار makefile قرار بگیرد (داخل پوشه‌ای فایل خروجی ساخته شده را قرار ندهید).
- درستی برنامه شما از طریق آزمون‌های خودکار سنجیده می‌شود؛ بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

# نمرات

ارزیابی پاسخ شما به تکلیف بر اساس موارد زیر انجام خواهد شد.

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و انسجام
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[6]
  - عدم استفاده از متغیرهای سراسری (گلوبال)
  - استفاده **مناسب** از متغیرهای ثابت به جای Magic Value-ها
- درستی کد
  - آزمون‌های خودکار
- طراحی
  - شکستن به کلاس‌های مناسب و تخصیص مسئولیت‌های صحیح به هر کلاس و استفاده از Header Guard-ها
  - جداسازی منطق کد از ورودی/خروجی
  - رعایت سطح دسترسی (public/private) در ویژگی‌های کلاس
  - عدم وجود منطق در تابع main
  - ساختاردهی کد در قالب توابع/متدهای کوتاه که فقط یک کار را انجام می‌دهند
- گیت و گیت‌هاب

[6] Indentation

- استفاده از commit messageهای مناسب
- هر کامیت یک کار مشخص انجام بدهد و کامیتی چندین کار انجام ندهد.

*دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.*