

## مجموعه ها

در این سوال قصد داریم کدی بنویسیم که مجموعه های ریاضی را پیاده سازی کند و حاصل عملیات های بین آنها رو محاسبه بنماید. تضمین می شود که اعضای مجموعه ها فقط int باشند.

کد شما باید قادر به انجام دستورات زیر باشد:

### تعریف یک مجموعه :

برای مثال با دادن ورودی زیر به برنامه، مجموعه A تعریف و ذخیره می شود

$$A = \{1, 2, 3\}$$

- نکته: لازم به ذکر نیست که در مجموعه ها تکرار اعضا وجود ندارد و در صورتی که ورودی دو عضو یکسان داشت، باید یکبار ذخیره شود.
- نکته: مجموعه ها را باید با یک حرف بزرگ نمایش داده شوند و تضمین می شود این قاعده در دادن ورودی رعایت شود.
- نکته: قرار دادن اسپیس های اضافی (و کم کردن اسپیس ها) نباید در کارکرد کد خلی ایجاد کند، برای مثال، کد شما باید تمام وروی های زیر را هندل کند.

$$A = \{ 1, 2, 3\}$$

$$A = \{1, 2 ,3 \}$$

$$A=\{1,2,3\}$$

در صورتی که مجموعه با موفقیت ساخته شد باید پیام زیر به کاربر نمایش داده شود :

set created successfully

در صورتی که مجموعه A قبل ساخته شده باشد کد شما باید مقادیر آن رو بر اساس ورودی جدید به روز کند و پیام زیر را به کاربر دهد:

```
set updated successfully
```

## چاپ یک مجموعه :

: ورودی

```
print B
```

در صورتی که مجموعه B قبل از تعریف شده باشد، باید اعضای آن را مشابه زیر چاپ کنید. توجه کنید که اعضا باید از کوچک به بزرگ مرتب شده باشند.

```
B = {3,4,5}
```

در صورتی که مجموعه B تعریف نشده باشد باید پیام خطای زیر را چاپ کنید:

```
B is not defined
```

## اضافه کردن یک عضو به یک مجموعه :

: ورودی

```
add 7 to C
```

در صورتی که مجموعه C تعریف نشده باشد باید پیام خطای زیر را چاپ کنید:

```
C is not defined
```

در صورتی که عدد 7 از قبل عضو C باشد باید پیام زیر چاپ شود:

```
7 is already in C
```

در صورت موفقیت پیام زیر به کاربر نمایش داده می‌شود

```
added successfully
```

## اعمال ساده بین مجموعه‌ها :

کد شما باید توانایی محاسبه اشتراک، اجتماع و تفاضل دو مجموعه را داشته باشد.

نکته: اشتراک را با  $*$  و اجتماع را با  $+$  نشان میدهیم

: ورودی

```
E = A*B  
print E
```

اشتراک A و B محاسبه می‌شود و حاصل در E ذخیر می‌شود. خروجی:

```
E = {3}
```

در صورتی که A یا B تعریف نشده باشند، پیام خطای زیر برای کاربر چاپ شود:

```
some sets are not defined
```

## چاپ زیر مجموعه‌های یک مجموعه:

: ورودی

```
subsets of A
```

کد شما باید تمام زیر مجموعه‌های مجموعه A را چاپ کند(ترتیب اهمیتی ندارد). مانند زیر:

```
{}
```

- {3}
- {2}
- {2,3}
- {1}
- {1,3}
- {1,2}
- {1,2,3}

درصورتی که مجموعه A تعریف نشده باشد باید پیام خطای زیر نشان داده شود:

B is not defined

## محاسبه عبارات جبر مجموعه ها

کدتان را به گونه ای توسعه بدھید که توانایی محاسبه عبارات پیچیده تری از جبر مجموعه ها را داشته باشد.

مثال :

عبارت 1:

$$D = A \cap (B \cup C)$$

ورودی متناظر با عبارت 1:

$$D = A^*(B+C)$$

عبارت 2:

$$G = (E \cup F) \cap ((A - B) \cup ((C \cap B) - D))$$

ورودی متناظر با عبارت 2:

$$G = (E+F)^*((A-B)+((C*B)-D))$$

این عبارات شامل پرانتز های تو در تو می باشند و کد شما باید حاصل نهایی را با رعایت کامل ترتیب عملیات ها و سایر قواعد جبر مجموعه ها محاسبه کند.

- نکته: در صورتی که یک یا چند تا از مجموعه های موجود در عبارت، تعریف نشده باشند، باید پیام زیر چاپ شود:

some sets are not defined

- نکته: همانطور که گفته شد، اشتراک با \* و اجتماع با + نشان داده می شود.
- نکته: تضمین می شود که ورودی، پرانتز گذاری مناسبی داشته باشد. برای مثال اگر اجتماع سه مجموعه A, B, C خواسته شود، به صورت زیر نوشته می شود:

$$D = A + (B + C)$$

در پایان با دستور end کار برنامه تمام می شود.

## زبان ساختاریافته کوئری

- محدودیت زمان: 5 ثانیه
- محدودیت حافظه: 128 مگابایت

هدف از این تمرین آشنایی اولیه با **Structured Query Language** و پیاده‌سازی یک نوع **RDBMS** ساده (که به صورت offline کار می‌کند) با زبان جاوا است. **SQL** یک زبان استاندارد برای ذخیره‌سازی، تغییر و دریافت داده‌ها در یک پایگاه داده (**Database**) است. باید بدانید این زبان در سازمان بین‌المللی استانداردسازی (ISO) ثبت شده است. وظیفه شما در این تمرین، نوشتن برنامه‌ای است که دستورات SQL را ورودی بگیرد و تغییرات لازم را بر روی پایگاه داده اعمال کند. اما قبل از آشنایی با SQL لازم است با ساختار یک دیتابیس آشنا شویم.

- اگر با مفاهیم دیتابیس و SQL آشنایی دارید می‌توانید مستقیم به سراغ خواسته‌های سوال بروید.

هر پایگاه داده از تعدادی جدول (**TABLE**) تشکیل شده است و با یک نام مشخص می‌شود. برای مثال جدول **مشخصات مشتریان** یک شرکت :

قسمتی از جدول **Customers** از پایگاهداده معروف Northwind

همچنین هر جدول از مفاهیمی به نام **Field** و **column** و **record** تشکیل شده است. اگر بخواهیم برای هر کدام تعریف درستی ارائه کنیم می‌توان گفت :

- Record:

هر ورودی که در یک جدول وجود دارد، یک record یا همان ردیف (row) محسوب می‌شود. آیا می‌دانید چرا به ورودی‌های جدول ردیف می‌گوییم؟ به این دلیل که اگر بخواهیم یک ورودی جدید را در جدولی وارد کنیم، باید آن را به صورت یک ردیف وارد کنیم!

- Column :

ستون‌ها در جدول، داده‌های عمودی هستند که اطلاعات یک دسته از فیلد‌ها را در خود دارند.

- Field :

همانطور که گفتیم هر جدول دارای تعدادی ستون و ردیف است. تلاقی هر ردیف و ستون را یک فیلد یا یک خانه از جدول گوییم.

تصویر زیر در درک تعاریف بالا به شما کمک می‌کند :

توضیح تصویر

حال به سراغ آشنایی با چند دستور ساده SQL می‌رویم :

## SELECT

دستور **SELECT** برای انتخاب داده‌ها از یک پایگاه داده مورد استفاده قرار می‌گیرد. داده‌هایی که با دستور **SELECT** برمی‌گردند در جدول نتایج به نام **result\_set** ذخیره می‌شوند و به عنوان خروجی نمایش داده می‌شوند. این دستور به دو شکل مورد استفاده می‌گردد :

```
SELECT * FROM table_name;
```

این دستور تمام داده‌های موجود از جدولی به نام **table\_name** را خروجی می‌دهد. برای مثال اگر از دستور زیر استفاده کنیم :

```
SELECT * FROM Customers;
```

باید تمام جدول مشتریان که در قسمت قبل دیدیم با تمام ردیف‌ها و ستون‌های آن به نمایش در بیاید :

از پایگاهداده معروف **Northwind** قسمتی از جدول **Customers**

شکل دیگر استفاده از این دستور به روشن زیر است :

```
SELECT column1,column2, ... FROM table_name;
```

با استفاده از این دستور تنها ستون‌های ذکر شده به همین اسم و به همین ترتیب در دستور، از جدولی به نام **table\_name** انتخاب و در جدول **result\_set** ذخیره و در خروجی نمایش داده می‌شوند. برای مثال اگر از دستور زیر استفاده کنیم :

```
SELECT City,CustomerName FROM Customers;
```

جدول زیر در خروجی نمایش داده می‌شود:

- در صورتی که جدولی به اسم ذکر شده، در دیتابیس ما وجود نداشت و یا اینکه اسمی ستون‌های ذکر شده در ورودی با اسمی ستون‌های جدول همخوانی نداشت؛ باید عبارت "ERROR!" در خروجی چاپ شود.
- دقت کنید که ترتیب ستون‌های result\_set به ترتیب اسمی ستون‌هایی هستند که در ورودی آمده است.
- تکرار اسمی ستون‌ها نباید مشکلی ایجاد کند و تکرار‌های اسم یک ستون در ورودی باید نادیده گرفته شوند به مثال زیر دقت کنید :

```
SELECT CustomerName,City,CustomerName,Country,Country,City FROM Customers;
```

## INSERT INTO

این دستور برای وارد کردن داده جدید به دیتابیس استفاده می‌شود. استفاده از آن به دو شکل صورت می‌گیرد :

```
INSERT INTO table_name VALUES (value1,value2,value3,...);
```

این دستور یک ریکورد(ردیف) جدید با مقادیر داده شده در ورودی را به جدولی به اسم table\_name اضافه می‌کند و در خروجی عبارت زیر چاپ می‌شود :

```
You have made changes to the database. Rows affected: 1
```

برای مثال اگر ورودی زیر را به برنامه بدهیم. یک ردیف به جدول مشتریان با مقادیر داده شده اضافه می‌شود.

```
INSERT INTO Customers VALUES ('93','Roham kaveie','Roham','Ekbatan','Tehran','
```

اگر از دستور SELECT برای نمایش جدول مشتریان استفاده کنیم می‌توانیم ریکورد اضافه شده را در ته جدول

پيدا کنيم :

```
SELECT * FROM Customers;
```

شكل ديگر دستور INSERT INTO به شكل زير استفاده ميگردد :

```
INSERT INTO Customers (CustomerID,CustomerName,City) VALUES ('94','Roham kavei
```

با استفاده از اين دستور يك ريكورد(رديف) با مقادير داده شده به ترتيب در ستونهاي ذكر شده فرارگرفته شده  
You have made changes to the database. Rows" و به جدول اضافه ميشود. و در خروجي عبارت "affected: 1  
چاپ ميشود.

اگر با دستور SELECT جدول را در خروجي چاپ کنيم ميتوانيم ريكورد اضافه شده را در خروجي ببینيم :

- همانطور که مشاهده ميکنيد در فيلدهايی که مقادير آنها داده نشد؛ مقدار null ذخیره شده است.
- اگر تعداد مقادير داده شده و اسمى ستونهاي داده شده با ستونهاي جدول همخوانی نداشته باشد  
باید عبارت "ERROR!" در خروجي چاپ شود.

## WHERE

از اين عبارت برای اضافه کردن شرط به دستور های ديگر SQL استفاده ميشود. به مثال زير دقت کنيد:

```
SELECT * FROM Customers WHERE Country = 'France';
```

این دستور در result\_set تنها ريكوردهايی که مقدار ستون Country آنان برابر با مقدار 'France' هستند را  
ذخیره و در خروجي نمایش ميدهد.

يا در مثال زير، دستور زير تنها ID مشترياني که شهر سکونت آنان پاريس است را به عنوان خروجي نمایش  
ميدهد :

```
SELECT CustomerID FROM Customers WHERE City = 'Paris';
```

- در زبان SQL می‌توانید با عبارات AND و OR و NOT و همینطور اپراتورهایی مانند <,>,=,<=,>=,!= شرایط پیچیده‌تری گذاشت و یا حتی با عبارت LIKE پترن هایی مشابه به ریجکس را برای پیدا کردن ریکورد های خاص استفاده کرد؛ اما برای سادگی در این تمرین تضمین می‌شود هیچکدام از آنها ورودی داده نمی‌شود و تنها ورودی‌هایی مشابه با دو مثال ذکر شده ورودی داده می‌شود.

## UPDATE

این دستور برای تغییر و بروزرسانی مقادیر ذخیره شده در پایگاه داده استفاده می‌شود. ساختار کلی این دستور به شکل زیر است :

```
UPDATE table_name SET column1 = value1,column2 = value2,... WHERE condition;
```

این دستور مقادیر خانه‌های ریکورد هایی که در شرایط condition صدق می‌کنند را بروزرسانی می‌کند و مقادیر ذکر شده را در ستون‌های گفته شده جایگذاری می‌کند. و عبارت زیر را در خروجی چاپ می‌کند:

```
You have made changes to the database. Rows affected: %d
```

- + عبارت %d تعداد ریکورد هایی است که در اثر این دستور تغییر پیدا کده‌اند.
- توجه کنید اگر از عبارت WHERE استفاده نشود تمام ریکورد های جدول انتخاب و آپدیت می‌شوند.

حال به مثال دیگری توجه کنید:

```
UPDATE Customers SET CustomerName = 'Arsham',ContactName = 'AK' WHERE Customer
```

همانطور که مشاهده می‌کنید تنها ریکوردی که شماره ID آن برابر با 93 بود مقادیر ستون‌های CustomerName و آن به مقادیر ذکر شده در ورودی تغییر پیدا کرد. ContactName

به مثال دیگری توجه کنید:

```
UPDATE Customers SET Country = 'Iran' WHERE Country = 'USA';
```

با اجرا این دستور تمام ریکوردهای جدول آنان برابر با 'USA' است با مقدار 'Iran' جایگذاری می‌شود. و خروجی زیر چاپ می‌شود :

```
You have made changes to the database. Rows affected: 13
```

و اگر پس از این دستور، دستور زیر را ورودی بدھیم با این خروجی مواجه می‌شویم :

```
SELECT * FROM Customers WHERE Country = 'Iran';
```

## DELETE

این دستور برای حذف داده‌ها در یک پایگاهداده استفاده می‌شود. ساختار کلی این دستور به شکل زیر است :

```
DELETE FROM table_name WHERE conditions;
```

این دستور تمام ریکوردهایی که در شرایط table\_name وجود دارند را حذف می‌کند و سپس عبارت زیر را به عنوان خروجی چاپ می‌کند :

```
You have made changes to the database. Rows affected: %d
```

- عبارت %d تعداد ریکوردهایی است که در اثر این دستور تغییر پیدا کردند.
- توجه کنید اگر از عبارت WHERE استفاده نشود تمام ریکوردهای جدول انتخاب و حذف می‌شوند.

به مثال زیر دقت کنید:

```
DELETE FROM Customers WHERE Country = 'UK';
```

پس از اجرای این دستور اگر با دستور SELECT ریکوردهایی که مقدار ستون Country آنان برابر با 'UK' است را خروجی نماییم. در خروجی تنها عبارت "No result" نمایش داده می‌شود (چون تمام ریکوردهایی با این مشخصه از قبل حذف شده‌اند) :

```
SELECT * FROM Customers WHERE Country = 'UK';
```

- توجه کنید در صورت حذف تمام ریکوردهای یک جدول، آن جدول خالی می‌شود اما حذف نمی‌شود و هنوز جدولی به این اسم در پایگاه داده ما موجود است. (صرفاً خالی از ریکورد است).

## CREATE TABLE

با این دستور می‌توانید یک جدول بسازید. ساختار کلی این دستور به شکل زیر می‌باشد :

```
CREATE TABLE table_name (column1,column2,column3,...);
```

با اجرای این دستور یک TABLE به اسم table\_name با ستون‌های column1,column2,...,columnn می‌شود و عبارت زیر به عنوان خروجی چاپ می‌شود:

```
You have made changes to the database.
```

لازم به ذکر است که جدول ساخته شده در ابتدا هیچ ریکوردی ندارد. برای مثال :

```
CREATE TABLE Company (CompanyID,CompanyName,Address,PhoneNumber,City,Country);
```

با اجرای این دستور یک جدول به اسم Company با ستون‌هایی که در ورودی ذکر شده است ساخته می‌شود. که اگر با دستور SELECT آن را نمایش دهیم عبارت "No result" نمایش داده می‌شود (چون هنوز هیچ ورودی به جدول داده نشده است).

اگر جدولی با این اسم قبلاً وجود داشته است یا اسامی ستون‌های ذکر شده در ورودی تکراری باشد باید عبارت

چاپ شود. !ERROR

## DROP TABLE

از این دستور برای حذف جدول‌های یک پایگاه داده استفاده می‌شود. ساختار کلی این دستور به شکل زیر است :

```
DROP TABLE table_name;
```

این دستور جدولی به اسم table\_name را از دیتابیس به طور کل حذف می‌کند و دیگر جدولی به این اسم نداریم. و در خروجی عبارت زیر را چاپ می‌کند:

You have made changes to the database.

برای مثال برای حذف جدول Company که در قسمت قبل ساختیم باید بنویسیم :

```
DROP TABLE Company;
```

- اگر جدولی به این اسم وجود نداشته باشد باید در خروجی عبارت "ERROR!" چاپ شود.

## show tables

شكل کلی این دستور به شکل زیر است :

```
show tables;
```

این دستور نام تمام جدول‌های داخل دیتابیس را با تعداد ریکورد هایی که دارند را به شکل زیر چاپ می‌کند :

Tablename	Records
table1	number1
table2	number2
...	

برای مثال برای دیتابیس northwind با استفاده از این ورودی با همچین خروجی مواجه می شویم :

Tablename		Records
Customers		91
Categories		8
Employees		10
OrderDetails		518
Orders		196
Products		77
Shippers		3
Suppliers		29

اگر هیچ جدولی در دیتابیس وجود نداشته باشد بیاد خروجی زیر چاپ شود :

No result.

این دستور در هر یک از سیستم های مدیریت دیتابیس (MySQL, SQL Server, SQLite,...) شکل منحصر به فرد خودش را دارد. می توانید [مطالعه](#) کنید.

## خواسته های سوال

از شما می خواهیم که ساختار یک دیتابیس را پیاده سازی کنید و برنامه خود را به گونه ای توسعه دهید که بتواند دستورات

۱. CREATE TABLE

۲. DROP TABLE

۳. INSERT INTO

۴. SELECT (WHERE)

۵. DELETE (WHERE)

۶. UPDATE (WHERE)

۷. show tables

را ورودی بگیرد و تغییرات لازم را بر اساس تعریف هر دستور بر روی دیتابیس اعمال کند.

به این منظور از شما میخواهیم که ابتدا در برنامه خود دو کلاس به اسمی InputProcessor و Manager وجود آورید. کلاس InputProcessor وظیفه گرفتن ورودی از کاربر و صدا زدن توابع لازم از Manager بر پایه ورودی داده شده و چاپ کردن خروجی لازم را بر عهده دارد.

در ادامه باید 3 کلاس به اسمی Database و Record و method با هایی که در classdiagram زیر مشخص شده‌اند به وجود آورید. ساختار هر کدام به شکل زیر می‌باشد :

## ورودی

در هر خط از ورودی یک دستور SQL به شما داده می‌شود. در آخرین خط ورودی به عبارت end خط می‌شود. که به معنای تمام شدن ورودی است.

```
CREATE TABLE Students (name,code,email,address);
SELECT * FROM Students;
INSERT INTO Students VALUES ('Arsham Kaveie','405000000','arsham@gmail.com','1
SELECT * FROM Students;
CREATE TABLE Universities (name);
show tables;
DROP TABLE Univer;
DROP TABLE Universities;
end
```

## خروجی

```
You have made changes to the database.
No result.
You have made changes to the database. Rows affected: 1
name | code | email | address
Arsham Kaveie | 405000000 | arsham@gmail.com | Tehran
You have made changes to the database.
Tablename | Records
Students | 1
```

Universities | 0

ERROR!

You have made changes to the database.

## \*نکات :

- هرگونه پیاده‌سازی ساختار کلاس‌های نامبرده شده که در ازای ورودی، خروجی درست و معتبر چاپ شود در صورت رعایت اصول شیگرایی و حفظ چارچوب‌های کلی یک پایگاه داده، قابل قبول است.
- برای چاپ کردن جدول‌ها ابتدا نام‌های هر ستون در سطر اول چاپ شود و سپس ریکوردهای جدول در سطرهای بعدی(هر کدام در یک سطر) چاپ شود. هر دو ستون عبارات با " | " از هم جدا شوند. (نمونه یک جدول چاپ شده در مثال ورودی و خروجی آمده است)
- تمرين به صورت داوری در کوئرا با تست کیس تصحیح می‌شود پس در نمایش خروجی‌های تعریف شده دقت کنید و دقیقاً مطابق حالت خواسته شده خروجی را چاپ کنید. (مورد قبلی)
- تضمين می شود** که اسمی جدول‌ها و ستون‌ها از اعداد و حروف لاتین (بزرگ یا کوچک) تشکیل شده باشند.
- تضمين می شود** که کوئری‌های تو در تو به عنوان ورودی داده نمی‌شود و در هر خط تنها یک دستور از 7 دستور گفته شده به شما داده می‌شود. نمونه یک کوئری تو در زبان SQL :

```
INSERT INTO Customers (CustomerName, City, Country) SELECT SupplierName, City,
```

- تضمين می شود که هر دستور با کاراکتر سمیکولن خاتمه پیدا کند.
- تضمين می شود که هر دستور جداگانه در یک خط ورودی داده شود و هر خط از ورودی شامل دستور معتبری باشد.
- زبان SQL به کوچک و بزرگ بودن حروف کلمات رزو شده (...,SELECT,UPDATE,...) حساس نیست با این حال **تضمين می شود** که تمام کلمات رزورشده با حروف بزرگ ورودی داده شوند.
- تضمين می شود** که کلمات رزور شده مانند SELECT,UPDATE,DELETE,... به عنوان اسمی جدول‌ها و ستون‌ها ورودی داده نشود.
- برای یادگیری و آشنایی بیشتر با SQL می‌توانید از لینک‌های زیر استفاده کنید. (برای علاقه‌مندان)

<https://www.roxo.ir/sql-language-introduction>

<https://www.sqltutorial.org>

<https://www.w3schools.com/sql/default.asp>

موفق باشید.

## دانشگاه شما دانشگاه نیست!

- محدودیت زمان: ۵۰ ثانیه
- محدودیت حافظه: ۵۰ مگابایت

بیگ مسعود بعد از اینکه تمرین سری یک را انجام داد، تصمیم گرفت تا انتقام مشکلات انتخاب واحد خود را از مسئولین آموزش بگیرد برای همین با تکنیکهای که بلد بود تلاش کرد تا به سامانه‌ی آموزش نفوذ کند اما سامانه‌ی آموزش با پیام "دانشگاه شما دانشگاه نیست" تحمل این نفوذ را نکرد و به طور کامل از دسترس خارج شد. مسئولین آموزش پس از اطلاع از این جریان تصمیم گرفتند به بیگ مسعود فرصت جبران بدهند (چرا که اگر او را اخراج می‌کردند میانگین معدل دانشگاه افت میکرد!!). بیگ مسعود پس از تفکرات فراوان برای این سامانه، توانست UML‌ای طراحی کند اما از آنجایی که این حجم از تفکرا باعث شد او از درس‌های دیگرش عقب بیفتند از شما به عنوان دوستش کمک خواسته تا این UML را به کد تبدیل کنید و خودش درس بتواند درس‌های دیگرش را بخواند. او در عوض به شما قول می‌دهد در صورت قبول این زحمت، نمره یک سوال از تمرین ۲ درس OOP را به شکل مخفیانه برای شما وارد سامانه کند.

Course
<u>- courses : ArrayList&lt;Course&gt;</u>
<u>- courseID : int</u>
<u>- unit : int</u>
<u>- capacity : int</u>
<u>- numberOfRegisteredStudents : int</u>
<u>- score : double</u>
<u>+ addNewCourse(course : Course) : void</u>
<u>+ getCourses() : ArrayList&lt;Course&gt;</u>
<u>+ getNumberOfRegisteredStudents() : int</u>
<u>+ setNumberOfRegisteredStudents(number : int) : void</u>
<u>+ incrementNumberOfRegisteredStudents() : void</u>
<u>+ decrementNumberOfRegisteredStudents() : void</u>
<u>+ getCourseID() : int</u>
<u>+ addCapacity(number : int) : void</u>
<u>+ getScore() : double</u>
<u>+ getUnit() : int</u>
<u>+ setScore(score : double) : void</u>
<u>+ getCapacity() : int</u>
<u>+ showCourse( courseID : int, string : String ) : void</u>

Student

```

- students : ArrayList<Student>
- studentID : int
- studentCourses : ArrayList<Course>
- average : double

+ getAverage() : double
+ addNewStudent(student : Student) : void
+ registerCourse(studentID : int, courseID : int) : void
+ addCourse(course : Course) : void
+ deleteCourse(StudentID : int, courseID : int) : void
+ setMark(courseID : int, mark : double) : void
+ setAverages() : void
+ getStudents() : ArrayList<Student>
+ getStudentID() : int
+ getStudentCourses() : ArrayList<Course>
+ deleteStudent(studentID : int) : void
+ showAverage(studentID : int) : void
+ showRanks() : void
+ showRanks(courseID : int) : void

```

### Lecturer

```

- lecturers : ArrayList<Lecturer>
- lecturerID : int
- courses : ArrayList<Course>

+ getLecturers() : ArrayList<Lecturer>
+ addNewLecturer(lecturer : Lecturer) : void
+ addCourse(courseID : int) : void
+ getCourses() : ArrayList<Course>
+ getLecturerID() : int
+ addCapacity(lecturerID : int, courseID : int, number : int) : void
+ setMark(lecturerID : int, courseID : int, mark : double, studentID : int) : void

```

**نکته:** تمامی کلاس‌ها باید مطابق با نمودار UML داده شده پیاده‌سازی شده شود. در صورتی که نیاز به متدهای کلاس کمکی که در UML ذکر نشده دارید، می‌توانید آن را پیاده‌سازی کنید اما چارچوب کلی که شما باید مبتنی بر UML داده شده باشد. در غیر این صورت بیگ مسعود اخراج شده و در نتیجه شما هم نمره‌ی سوال ۱ تمرین ۲ این درس را نمی‌گیرید!

## ویژگی‌های سامانه‌ی آموزش:

`addStudent <sID>`

- دانشجویی با شناسه‌ی sID به سامانه اضافه می‌شود.

`addLecturer <ltID> <cID1> <cID2> ....`

- استاد با شناسه‌ی tID به سامانه اضافه می‌شود. این استاد دروس cID1، cID2 و ... را ارائه می‌دهد.  
همچنین ممکن است یک استاد هیچ درسی ارائه نکند یا فقط یک درس ارائه کند.

`W <cID> <sID>`

- دانشجوی با شماره‌ی sID درس cID را حذف می‌کند.

`<sID> register <cID>`

- دانشجوی با شماره‌ی sID در درس cID ثبت‌نام می‌کند.

`<sID> register <cID1> <cID2> ...`

- دانشجوی با شماره‌ی sID در درس‌های cID1 و cID2 و ... ثبت‌نام می‌کند.

`<ltID> capacitate <cID> <n>`

- استاد با شناسه‌ی ltID ظرفیت درس cID را به اندازه‌ی n افزایش می‌دهد.

`<ltID> mark <cID> <sID1> <mark1> <sID2> <mark2> ...`

- استاد با شناسه‌ی ltID که درس cID را ارائه می‌دهد، برای دانشجوی sID1 نمره‌ی mark1، برای دانشجوی

sID نمره‌س mark2 و ... را ثبت می‌کند. تضمین می‌شود نمره‌ی تمام دانشجویان درس cID وارد می‌شود.

<ltID> mark <cID> <mark> -all

- استاد با شناسه‌ی ltID برای دانشجویان درس cID نمره‌ی mark را رد می‌کند.

start semester

- شروع ترم جاری

end semester

- پایان ترم جاری

end registration

- پایان زمان ثبت‌نام

addCourse <sID> <unit>

- درس با شناسه‌ی cID که واحد دارد به سامانه اضافه می‌شود.

### **:show دستورات**

showCourse <cID> <students|lecturer|capacity|average>

- برای درس cID ویژگی خواسته شده نمایش داده می‌شود. (اگر students بود لیست تمامی دانشجویان ثبت‌نامی به ترتیب شماره‌ی دانشجویی، اگر lecturer بود، شناسه‌س استاد ارائه‌دهنده‌ی درس، اگر capacity بود، ظرفیت درس و اگر average بود، معدل درس را چاپ کند) در صورت معتبر نبودن شماره‌ی درس یا حذف شدن آن (به حد نصاب نرسیدن)، پیغام shoma daneshjoo nistid را چاپ کنید.

`showRanks <sID>`

- شناسه‌ی سه دانشجویی که بالاترین نمرات درس را کسب کرده‌اند چاپ می‌کند. (در صورت برابر بودن نمره‌ی چند دانشجو، سه نفر اول به ترتیب شماره‌ی دانشجویی چاپ شوند)

`shawAverage <sID>`

- معدل یک دانشجو را چاپ می‌کند. در صورت وجود نداشتن دانشجوی با شناسه‌ی sID، پیام shoma nistid را چاپ کنید.

`showRanks -all`

- نمایش دانشجویان با رتبه‌ی 1 تا 3 از نظر معدل. (در صورت برابر بودن، به ترتیب شماره‌ی دانشجویی چاپ شوند)

نکات دیگر:

- هر درس به طور پیش فرض دارای 15 نفر ظرفیت می‌باشد و باید دارای حداقل 3 نفر برای تشکیل باشد.
- هر دانشجو باید حداقل 12 واحد در یک ترم داشته باشد.
- هر درس تنها توسط یک استاد رائه می‌شود.
- هر دانشجو تنها 3 واحد را می‌تواند W کند به این صورت که تا زمانی که واحدهایی که درخواست W آن‌ها را داده از 3 واحد بیشتر نیستند، آن‌ها حذف می‌شوند و سپس دیگر حذف نمی‌شوند. مثلاً اگر دستور حذف یک درس 4 واحدی را وارد کند، نه تنها درس حذف نمی‌شود بلکه هیچ واحد دیگری نیز نمی‌تواند حذف کند.
- شناسه اساتید و دانشجویان 5 رقمی هستند.
- با شروع ترم، ثبت‌نام نیز شروع می‌شود.
- پس از پایان زمان ثبت‌نام، ابتدا تمام دروسی که به حد نصاب نرسیده‌اند حذف شده و دانشجویان دیگر آن درس‌ها را نخواهند داشت. سپس تعداد واحد تمام دانشجویان چک می‌شود. پس از این مرحله دیگر

درس‌ها چک نشده و ترم رسمای آغاز می‌گردد.

- در صورتی که یک دانشجو از کف مجاز واحد کمتر داشته باشد، ترم وی حذف می‌شود.
- دستورات اضافه شدن دانشجو و استاد قبل شروع ترم وارد می‌شوند.
- پس از اتمام ترم دستورات `show` وارد می‌شوند و برنامه با دستور `endShow` پایان می‌پذیرد.
- در صوت وارد شدن دستورات بی‌ربط چیزی چاپ نمی‌شود.
- تمامی محاسبات اعشاری با دقت یک رقم اعشار محاسبه شود.

## ورودی نمونه ۱

```
addCourse 40101 3
addCourse 40102 4
addCourse 40103 4
addCourse 40104 3
addCourse 25001 4
addCourse 25002 3
addCourse 25003 4
addCourse 65001 3
addCourse 65002 3
addCourse 65003 4
addStudent 95101
addStudent 95102
addStudent 95103
addStudent 95103
addStudent 95104
addStudent 94101
addStudent 94102
addStudent 94103
addLecturer 10001 40101 40103
addLecturer 10002 40102 40104
addLecturer 10003 25001 25002
addLecturer 10004 65001 65002 65003
start semester
95101 register 40101 40102 40103 25001 25002
95102 register 40101 40102 40103 25001 25002
95103 register 40101 40102 40103 25001 25002
95104 register 40101 40102 40103 25002 65002
```

```

94101 register 40102 40103 25001 25002 65001
94102 register 40102 40103 25001 25002 65001
94103 register 40102 40103 25001 25002 65002
end registration

10001 mark 40101 95101 12 95102 18 95103 18.5 95104 9.9
10002 mark 40102 95101 19.8 95102 20 95103 16.5 95104 19 94101 17 94102 10 941
10001 mark 40103 95101 17.4 95102 19 95103 15.7 95104 11.2 94101 19 94102 17.8
10003 mark 25001 95101 19 95102 20 95103 18.4 95104 13.3 94101 18 94102 15.2 9
10003 mark 25002 95101 20 95102 14 95103 18 95104 9.5 94101 20 94102 18 94103
10004 mark 65001 94101 12 94102 20
end semester

showCourse 40101 lecturer
showCourse 40102 average
showCourse 40103 average
showCourse 65001 lecturer
showAverage 95101
showAverage 95102
showRanks 25001
endShow

```

### خروجی نمونه ۱

```

10001
16.9
17.2
shoma daneshjoo nistid
17.8
18.4
95102 95101 95103

```

### ورودی نمونه ۲

```

addCourse 40101 12
addCourse 40102 12
addCourse 40103 12
addStudent 95101
addStudent 95102

```

```
addStudent 95103
addStudent 95104
addStudent 95105
addStudent 95106
addStudent 95107
addStudent 95108
addStudent 95109
addStudent 95110
addStudent 95111
addStudent 95112
addStudent 95113
addStudent 95114
addStudent 95115
addStudent 95116
addStudent 95117
addStudent 95118
addStudent 95119
addStudent 95120
addStudent 95121
addStudent 95122
addStudent 95123
addStudent 95124
addLecturer 10001 40101 40102 40103
start semester
10001 capacitate 40102 50
95101 register 40101 40102
95102 register 40101 40102
95103 register 40101 40102 40103
95104 register 40101 40102
95105 register 40101 40102
95106 register 40101 40102 40103
95107 register 40101 40102
95108 register 40101 40102
95109 register 40101 40102 40103
95110 register 40101 40102
95111 register 40101 40102
95112 register 40101 40102 40103
95113 register 40101 40102
95114 register 40101 40102
95115 register 40101 40102 40103
```

```

95116 register 40101 40102
95117 register 40101 40102
95118 register 40101 40102 40103
95119 register 40101 40102
95120 register 40101 40102
95121 register 40101 40102 40103
95122 register 40101 40102
95123 register 40101 40102
95124 register 40101 40102 40103
end registration
10001 mark 40101 20 -all
10001 mark 40102 15 -all
10001 mark 40103 10 -all
end semester
showCourse 40101 students
showCourse 40102 capacity
showAverage 95111
showAverage 95123
endShow

```

## خروجی نمونه ۲

```

95101 95102 95103 95104 95105 95106 95107 95108 95109 95110 95111 95112 95113
65
17.5
15

```

## ورودی نمونه ۳

```

addCourse 40101 3
addCourse 40102 4
addCourse 40103 4
addCourse 40104 3
addCourse 25001 4
addCourse 25002 3
addCourse 25003 4
addCourse 65001 3

```

```
addCourse 65002 3
addCourse 65003 4
addStudent 95101
addStudent 95102
addStudent 95103
addStudnet 95103
addStudent 95104
addStudent 94101
addStudent 94102
addStudent 94103
addLecturer 10001 40101 40103
addLecturer 10002 40102 40104
addLecturer 10003 25001 25002
addLecturer 10004 65001 65002 65003
start semester
95101 register 40101 40102 40103 25001 25002
95102 register 40101 40102 40103 25001 25002
95103 register 40101 40102 40103 25001 25002
95104 register 40101 40102 40103 25001 25002 65002
94101 register 40102 40103 25001 25002 65001
94102 register 40102 40103 25001 25002 65001
94103 register 40102 40103
end registration
10001 mark 40101 95101 12 95102 18 95103 18.5 95104 9.9
10002 mark 40102 95101 19.8 95102 20 95103 16.5 95104 19 94101 17 94102 10 941
10001 mark 40103 95101 17.4 95102 19 95103 15.7 95104 11.2 94101 19 94102 17.8
10003 mark 25002 95101 19.8 95102 14 95103 18.3 95104 9.5 94101 20 94102 18
10003 mark 25001 95104 11 94101 13.1 94102 13 95101 9 95103 20 95102 18
end semester
showAverage 94103
showAverage 94101
showRanks 65001
showRanks 25002
showRanks -all
endShow
```

خروجی نمونه ۳

```
shoma daneshjoo nistid  
17.1  
94101 95101 95103  
95102 95103 94101
```

## ورودی نمونه ۴

```
addCourse 40101 3  
addCourse 40102 4  
addCourse 40103 4  
addCourse 40104 3  
addCourse 25001 4  
addCourse 25002 3  
addCourse 25003 4  
addCourse 65001 3  
addCourse 65002 3  
addCourse 65003 4  
addStudent 95101  
addStudent 95102  
addStudent 95103  
addStudent 95103  
addStudent 95104  
addStudent 94101  
addStudent 94102  
addStudent 94103  
addLecturer 10001 40101 40103  
addLecturer 10002 40102 40104  
addLecturer 10003 25001 25002  
addLecturer 10004 65001 65002 65003  
start semester  
95101 register 40101 40102 40103 25001 25002  
95102 register 40101 40102 40103 25001 25002  
95103 register 40101 40102 40103 25001 25002  
95104 register 40101 40102 40103 25002 65002  
94101 register 40102 40103 25001 25002 65001  
94102 register 40102 40103 25001 25002 65001  
94103 register 40102 40103 25001 65001  
end registration
```

```

10001 mark 40101 95101 12 95102 18 95103 18.5 95104 9.9
10002 mark 40102 95101 19.8 95102 20 95103 16.5 95104 19 94101 17 94102 10 941
10001 mark 40103 95101 17.4 95102 19 95103 15.7 95104 11.2 94101 19 94102 17.8
10003 mark 25001 95101 9 95102 10 95103 10 94101 10 94102 19 94103 12
10003 mark 25002 95101 20 95102 14 95103 18 95104 9.5 94101 20 94102 18 94103
10004 mark 65001 94101 18 94102 10 94103 15
W 40101 95101
end semester
showAverage 95103
showCourse 25001 average
showRanks 65001
endShow

```

## خروجی نمونه ۴

```

15.5
11.7
94101 94103 94102

```

## ورودی نمونه ۵

```

addCourse 40101 3
addCourse 40102 6
addCourse 40103 4
addStudent 95101
addStudent 95102
addStudent 95103
addStudent 95103
addStudent 95104
addStudent 94101
addStudent 94102
addStudent 94103
addLecturer 10001 40101 40103
addLecturer 10002 40102 40104
addLecturer 10003 25001 25002
addLecturer 10004 65001 65002 65003
start semester

```

```
95101 register 40101 40102 40103 25002
95102 register 40101 40102 40103 25002
95103 register 40101 40102 40103 25002
95104 register 40101 40102 40103 25002 65002
94101 register 40102 40103 25001 25002 65002
94102 register 40102
94103 register 40102 40103 25002 65002
10001 mark 40101 95101 0 95102 0 95103 0 95104 0
end registration
10001 mark 40101 95101 19 95102 20 95103 12 95104 10
10001 mark 40103 9.9 -all
10002 mark 40102 9.9 -all
10002 mark 40104 9.9 -all
10003 mark 25001 9.9 -all
10003 mark 25002 9.9 -all
10004 mark 65001 9.9 -all
10004 mark 65002 9.9 -all
10004 mark 65003 9.9 -all
W 40101 95103
end semester
showCourse 40101 average
showAverage 95103
showCourse 40102 lecturer
showRanks 40101
endShow
```

## خروجی نمونه ۵

```
16.3
9.9
10002
95102 95101 95104
```

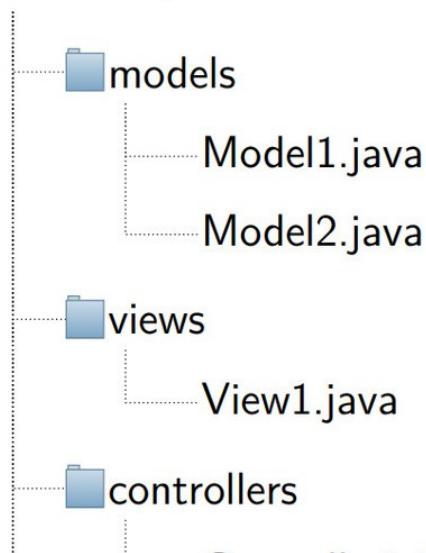
توجه داشته باشید که در این سوال محدودیت زمانی دارید و از سریع ترین روش‌ها استفاده کنید.

این سوال توسط سامانه کوئرا داوری و نمره‌دهی خواهد شد. توجه داشته باشید به هیچ‌وجه تصحیح این سوال

به صورت دستی انجام نخواهد شد. لطفا به موارد زیر توجه کنید:

- با توجه به اینکه SDK کوئرای نسخه 1.8 می‌باشد لطفا از نسخه استفاده کنید.
- در کل پروژه فقط یک شی Scanner داشته باشید.
- جهت آپلود کدهای خود به یکی از دو روش زیر اقدام کنید:
  - برای آپلود همه‌ی کلاس‌هارو کپی کنید زیر یک کلاس (توی یک فایل java) فقط توجه کنید که توی یک فایل فقط می‌توانید یک کلاس public داشته باشید. بعد از کپی کردن کلاس‌ها توی یک فایل پیشوند پابلیک کلاس‌هایی که کپی کردید رو بردارید تا فقط یک کلاس پابلیک باقی بماند.  
public static void main(string[] args) (حتما کلاسی که حاوی تابع public static void main(string[] args) بصورت باشد)
  - یک نمونه مثال ساده برای اینکه چندتا کلاس توی یک فایل داشته باشید.
- شما باید تمامی فایل‌های کلاس‌هایی که با فرمت zip نوشته‌اید را در قالب فشرده با فرمت zip درآورده و در سامانه کنید. توجه کنید که حتما فایل حاوی کلاسی که تابع public static void main(string[] args) در آن قرار دارد و اجرای برنامه از آن‌جا شروع می‌شود، باید بدون قرار گرفتن در هیچ پوشه‌ای درون فایل zip قرار بگیرد و تام آن هم Main.java باشد. امکان پوشه بندی و ایجاد package مختلف برای سایر فایل‌ها امکان پذیر است. به عنوان نمونه می‌توانید src که فایل‌ها در آن قرار دارد، نباید در فایل zip باشد.

answer.zip



```
.....Controller1.java  
.....Main.java  
....MyClass.java  
....MyClass2.java
```

- ورودی و خروجی شما باید عیناً شبیه به نمونه‌های ورودی و خروجی باشد؛ لذا عبارت‌هایی همچون  
را قبل از گرفتن ورودی نباید چاپ کنید.  
Enter your number

## آرایه پویا

برای مشاهده صورت متن سوال روی گزینه PDF کلیک کنید.

## تقلب یا ب

همانطور که می دانید از وقتی که کرونا وارد دنیا شده، انواع شیوه های تقلب در همه ی سطح ها گرفته از امتحانات مدرسه، آزمون های آزمایشی کنکور، امتحان دانشجو ها تا سطوح بالاتر به وجود آمده است. حالا در این سوال ما می خواهیم ابزاری طراحی کنیم که با کمک اطلاعات موجود و تحلیل برخی داده ها شامل افراد، گروه های تقلب و امتحانات برگزار شده، افراد خطا کار را شناسایی کنیم. اول از همه سامانه ی ما در صفحه ی اصلی قرار دارد که در این صفحه می توان عملیات های login یا signup را انجام داد دستور signup به شکل زیر وارد میشود:

```
register [username] [password] [teacher|student]
```

همانطور که مشخص است، یک کاربر با نام کاربری و پسورد مشخص شده را ایجاد می کند. نام کاربری و پسورد باید فقط شامل حروف الفبای انگلیسی، اعداد و کاراکتر آندرلاین باشند. خطاهای مربوط به این دستور به این ترتیب چک می شوند و هر خطا که رخ بدهد، پیغام مربوط به همان خطا چاپ شده و سایر خطاهای بررسی نمی شوند. اگر هم هیچ خطایی رخ نداد و عملیات موفقیت آمیز بود پیغام زیر چاپ می شود

```
register successful
```

خطاهای: اگر نام کاربری شامل کاراکترهایی به جز کاراکتر های ذکر شده بود، پیغام:

```
username format is invalid
```

اگر پسورد شامل کاراکتر هایی به جز کاراکتر های ذکر شده بود، پیغام:

```
password format is invalid
```

اگر کاربری با username گفته شده از قبل وجود داشت پیغام:

a user exists with this username

دستور بعدی ، دستور

login [username] [password]

برای لاگین کردن به حساب کاربری مشخص با نام کاربری و پسورد داده شده استفاده می شود. نام کاربری و پسورد باید فقط شامل حروف الفبای انگلیسی، اعداد و کاراکتر آندرلاین باشد. خطاهای مربوط به این دستور به این ترتیب چک می شوند و هر خطا که رخ داده بود، پیغام همان خطا چاپ شده و سایر خطاهای بررسی نمی شوند. اگر هم هیچ خطایی رخ نداد و وضعیت موفقیت آمیز بود، پیغام

login successful

چاپ خواهد شد و پس از آن کاربر به طور خودکار به صفحه ای اصلی خود وارد می شود

خطاهای:

اگر نام کاربری شامل کاراکتر هایی به جز کاراکترهای ذکر شده بود، پیغام:

username format is invalid

اگر پسورد شامل کاراکتر هایی به جز کاراکترهای ذکر شده بود، پیغام:

password format is invalid

اگر کاربری با username وارد شده وجود نداشت پیغام:

no user exists with this username

گر پسورد غلط بود پیغام:

incorrect password

چاپ خواهد شد

و بعد از ورود به منوی اصلی می توان داده هایی را وارد کرد و خواسته ها را نسبت به آن ها صادر کرد. با دستور

logout

هم میتوان از حساب کاربری خارج شد و خروجی این دستور هم جمله‌ی

logout successful

می باشد

دقیقت داشته باشید خروج از برنامه و در واقع پایان یافتن برنامه تنها از منوی ثبت نام و ورد امکان پذیر است و با دستور

exit

این اتفاق می افتد و برنامه تمام می شود . برای حل مسئله برخی فرض هارا در نظر می گیریم که شامل موارد زیر هست.

1-هر امتحان با آیدی ای که از یک شروع می شود، مشخص می شود.

2-هر فرد هم با آیدی که از یک شروع می شود، مشخص می شود.

3-هر فرد میتواند در هیچ، یک یا چند گروه تقلب باشد.

4-اگر فردی به عنوان مثال در دو گروه تقلب باشد، وقتی تقلب کند، کل اعضای دو گروهی که آن شخص در آن ها عضو است هم پایشان گیر است و در حالت مشکوک قرار می گیرند.

5- تاریخ آزمون و تاریخ زمانی که فردی تقلب می کند هم به فرمت yyyy/mm/dd داده میشود.

6- این موضوع را حتما می دانید ولی به عنوان تذکر ذکر می کنیم که تعداد روز ماه های میلادی به ترتیب از ژانویه تا دسامبر برابر با ۳۱، ۲۹، ۳۰، ۳۱، ۳۰، ۳۱، ۳۰، ۳۱، ۳۰، ۳۱ و ۳۱ است.

7- خیلی از اوقات پیش می آید که امتحان شخصی پشت سر هم باشد یعنی بدون فاصله بعد از یک امتحان، به محل برگزاری امتحان بعدی می رود و یا حتی در حین برگزاری یک امتحان، اجازه خواسته و به دستشویی یا بالای سطل آشغال برای تراشیدن مدادش یا انداختن آشغالی درون سطل آشغال می رود. پس ما یک سری جابه جایی در حین امتحان داریم که یا درون امتحانی است و یا بروون امتحانی. در این موارد این انتقال به صورت ورودی داده میشود و آیدی امتحان مبدا و مقصد مشخص می شود و در ادامه گفته می شود که خواسته سوال از شما در برخورد با این انتقالات به چه شکل باشد

8- در مورد قبل اگر جا به جایی درون امتحانی باشد، آیدی مبدا و مقصد یکسان می باشد.

9- در صورتی که شخصی در روز  $x$  ام تقلبی انجام دهد، تا روز  $x+7$  ام در حالت مشکوک به تقلب باقی ما ماند و در ادامه گفته می شود که در صورت انجام چه حرکتی، تقلب او لو می رود.

10- حالدرصورتی تقلب شخص لومی رودکه در بازه  $i\text{--}i+8$  روزه ای که مشکوک به تقلب هست، جا به جایی بین امتحان ها داشته باشد. یعنی اگر در یک امتحان تقلبی انجام دهد و در بازه  $i\text{--}i+8$  روزه در امتحان دیگری شرکت کند، تقلب او بر همگان آشکار میگردد.

11- دقیقت کنید جا به جایی درون امتحانی تخلف محسوب نمی شود و فرد مشکوک در این حالت لو نمی رود.

## 9 ردی

دقیقت کنید دستورات زیر فقط زمانی قابل اجرا هستند که شخص دارای role دانشجو باشد که در زمان ثبت نام برای او قرار داده شده است و اگر استادی بخواهد این دستورات را وارد کند باید برای او برگردانید

همه های ورودی زیر بعد از لاگین معتبر هستند و منظور از خط اول یا `n` ام در این ورودی ها، بعد از دستورات مرتبه با لاگین است.

- در خط اول ورودی  $n$  می‌آید که همان افراد مورد بررسی است یعنی افراد از آیدی ۱ تا  $n$

n

- در خط دوم  $m$  می‌آید که تعداد گروه‌های تقلب موجود هست

m

- بعد در  $m$  خط بعد اعضای گروه تقلب  $a$  ام با یک فاصله بین هم داده می‌شود

- بعد  $c$  داده می‌شود که تعداد امتحانات است و در واقع این یعنی امتحانات ما آیدی بین ۱ تا  $c$  دارند

c

- بعد  $p$  وارد می‌شود که در واقع تعداد جایی‌های انجام شده در دانشگاه است

p

- در  $p$  خط بعد اول آیدی فرد جا به جا شده می‌آید و بعد آیدی امتحان مبدا و بعد آیدی امتحان مقصد و در

نهایت هم تاریخ این جا به جایی وارد  م شود به فرمت YYYY/MM/DD

```
personId sourceExam destinationExam date
```

- سپس  $t$  وارد می‌شود که تعداد تلاش‌ها برای تقلب است

t

- در خط بعد، در هر خط اول آیدی شخصی که تلاش به تقلب کرده وارد می‌شود و بعد نتیجه‌ی تقلب وی داده می‌شود که در دو حالت `positive` و `negative` است که اگر `positive` بباید یعنی وی موفق به تقلب شده است و در غیر این صورت یعنی موفق به تقلب نشده است. و بعد آیدی امتحانی که آن شخص در آن تلاش به تقلب کرده است داده می‌شود و در نهایت هم تاریخ روزی که آن شخص خواسته تقلب کند به همان فرمت `/YYYY/MM/DD`

mm/dd داده میشود

personId cheatResult examId date

دقت کنید دستوراتی که چند بخشی هستند که شامل مشخص کردن اعضای هر گروه تقلب، جا به جایی های صورت گرفته توسط افراد و تلاش ها برای تقلب می شوند، تضمین می شود که معتبر هستند و نیازی نیست شما از صحبت ورودی ها اطمینان حاصل کنید ولی در هر دستور دیگری که ذکر نشد، شما

باید از صحبت فرمت ورودی اطمینان حاصل کنید و اگر معتبر نبود، پیام invalid command چاپ کنید. بعد از اینکه این دستورات وارد شد، یک round برای دانشجوی لایکین شده تمام می شود و در این قسمت، ۲ کارمی تواند انجام دهد، یا round بعدی را با وارد کردن nextround شروع کند و یا اینکه با زدن دستور logout، از پنل خود خارج شود. در صورتی که round بعدی را شروع کند، اطلاعاتی که وارد می کند، در ادامه ای اطلاعات قبلی م [!OF C00] یاًید، به عنوان مثال اگر در round اول در خط اول عدد ۶ را وارد کرده باشد، یعنی ۶ دانشجو از آیدی ۱ تا ۶ داریم و بعد ادامه اطلاعات را وارد می کند و در round بعدی اگر عدد مثلا ۳ را وارد کند یعنی دانشجویان جدیدی از آیدی ۷ تا ۹ داریم و در اطلاعات بعدی مثلا اعضای هر گروه، میتواند آیدی هر دانشجویی با ایدی ۱ را ئارد کند ۷ یعنی آن ۳ در خط اول تعداد دانشجویان جدید است و از بعد ان ایدی فرد جدید، ۷ تا ۹ است و نه ۱ تا ۳ همینطور برای امتحان ها هم به همین شکل که برای دانشجویان بوده عمل کنید. به این نکته هم دقت کنید که اگر یک دانشجو login کند و بعد دانشجوی دیگری logout کند و اطلاعاتی را وارد کند، آن اطلاعات هم در ادامه ای اطلاعات قبلی می آید

## درخواست ها

درخواست ها دقت کنید این درخواست ها فقط زمانی قابل وارد شدن هستند که شخصی که لایکین کرده دارای role استاد باشد و اگر آن دانشجو باشد، در هنگام وارد کردن این درخواست ها با invalid command مواجه می شود

- خواسته ها وارد می شود که پایین تر توضیحات آن ها را می توانید ملاحظه کنید و این وارد شدن دستورات تا زمان وارد شدن دستور clear data و یا end و یا logout ادامه پیدا می کند. دقت کنید در صورت وارد شدن دستور clear data کل اطلاعات به شکل کامل پاک می یشود و همه چیز از نو شروع می شود و در صورت وارد شد

دستور `end` بدون اینکه اطلاعاتی پاک شود، صرفاً گرفتن درخواست پایان می‌باید و با وارد شدن دستور `logout` از حساب کاربری استاد خارج می‌یشویم. اگر `logout` وارد شود که به منوی ثبت نام و ورود بر می‌گردیم ولی اگر `end` وارد شد در حالت انتظار می‌مانیم که یا می‌توان باز `logout` کرد و از حساب کاربری خارج شد و یا می‌توان یکی از دستورهای `wrist-up` را وارد کرد و به چرخه‌ی گرفتن دستور بر گردیم

#### 1. درخواست `wrist-up1` :

با این درخواست از شما می‌خواهیم که افراد با بیشترین تخلف ثبت شده را به ترتیب نزولی برای ما برگردانید

#### `wrist-up1 count`

که در واقع `count` تعداد افرادی است که در خروجی باید نمایش دهیم و در حالتی که تعداد تخلف ثبت شده دو نفر با هم برابر بود، فرد با آیدی کوچک تر، بالاتر قرار می‌گیرد. ما این دستور را به شکل دیگری هم از شما می‌خواهیم و آن به این شکل است که تاریخ شروع و پایان را هم در دستور به شکل زیر می‌دهیم و شما باید افراد با بیشترین تخلف ثبت شده در آن بازه زمانی را به ترتیب نزولی برای ما پیدا کنید. در این بخش هم فرمت تاریخ به شکل `yyyy/mm/dd` داده می‌شود

#### `wirst-up1 count startDate endDate`

2. درخواست `wrist-up2` : در این درخواست از شما می‌خواهیم که افراد با بیشترین تقلب `positive` شده را به ما نشان دهیم. دقت کنید در این حالت تقلب `positive` شده‌ی خود فرد منظور است و اگر یکی از هم گروهی هایش لو رفته باشد و این شخص را هم لو داده باشد، شما در شمارش این بخش نباید آن را وارد کنید و فقط تقلب `positive` شده‌ی خود آن شخص را در نظر بگیرید. و باز هم به شکل نزولی نمایش دهیم و در صورت برابر شدن تعداد تقلب‌های `positive` شده‌ی دو نفر، فرد با آیدی کوچک تر، بالاتر قرار بگیرد

#### `wrist-up2 count`

که در اینجا هم `count` تعداد افرادی هست که شما به ما باید نشان بدهیم. این دستور هم به شکلی دیگر می-

تواند وارد شود و آن هم به این شکل هست که تاریخ شروع و پایان بازه‌ی مورد نظر ما وارد می‌شود و شما افراد با بیشترین تقلب positive شده در آن بازه زمانی را برای ما باید برگردانید. در اینجا هم مانند بقیه بخش‌ها تاریخ با فرمت yyyy/mm/dd وارد می‌شود

wrist-up2 count startDate endDate

۳. درخواست wrist-up3 : در این درخواست از شما می‌خواهیم امتحان‌هایی با بیشترین تقلب positive شده در آن را برای ما نشان دهید. در اینجا هم دقت کنید که گفته شد امتحاناتی که بیشترین تقلب positive شده در آن رخ داد و این یعنی آن شخص خودش تقلیبش positive شده باشد و نه اینکه توسط گروهش لو داده شده باشد. در اینجا هم آن امتحان‌ها را به ترتیب نزولی نمایش دهید

wrist-up3 count

که در اینجا هم count تعداد امتحاناتی است که شما باید در خروجی برای ما برگردانید و در صورت برابر بودن تعداد افرادی که تقلب آن‌ها positive شده در دو امتحان، امتحان با آیدی کوچک‌تر، بالاتر قرار می‌گیرد. این دستور هم مانند دستور‌های قبل به این نوع هم می‌تواند وارد شوند که ما تاریخ شروع و پایان بازه‌ی مورد نظر خود را بگوییم و شما امتحاناتی که در آن بازه بیشترین تقلب positive شده را داشتند را برگردانید. و اینجا هم تاریخ با فرمت yyyy/mm/dd به شما داده می‌شود

wrist-up3 count startDate endDate

۴. درخواست wrist-up4 : در این درخواست، آیدی یک امتحان را به شما می‌دهیم و روز‌هایی که بیشترین تقلب positive شده در آن رخ داده را به ما نشان می‌دهید و به ترتیب نزولی در خروجی بر می‌گردانید. در این بخش شما تاریخ را باید به فرمت yyyy/mm/dd به ما بدهید

wrist-up4 examId count

که در اینجا هم count تعداد روزهایی است که شما در خروجی باید نمایش دهید و در صورتی که در دو روز

تعداد تقلب های positive شده برابر بود، روز قبل تر در خروجی، بالاتر قرار می گیرد.

۵. درخواست wrist-up5 : در این درخواست از شما می خواهیم روز هایی از سال با بیشترین تعداد افراد مشکوک به تقلب را به شکل نزولی در خروجی چاپ کنید. به این نکته ی خیلی مهم توجه کنید، همانطور که گفته شد در هر روزی که شخصی تقلب کند، خود آن روز و تا ۷ روز بعد از آن در حالت مشکوک به تقلب هست

wrist-up5 count

که باز هم count تعداد روز هایی است که در خروجی باید برگردانید و در صورت برابر بودن تعداد افراد مشکوک به تقلب در یک روز، روز قبل تر، بالاتر قرار می گیرد. و در اینجا هم فرمتی که ما تاریخ را از شما می خواهیم به شکل

yyyy/mm/dd

است

## خروجی

برای هر درخواستی که داده میشود در  خط شما اسم در خواست را چاپ می کنید یعنی یکی از مقادیر

wrist-up5 or wrist-up4 ,wrist-up3 ,wrist-up2 ,wrist-up1

و سپس نوع خروجی هر دستور متفاوت خواهد بود و به شکل زیر است:

- خروجی درخواست wrist-up1 count شامل خط است و در هر خط اول آیدی فردی که تقلب اون آشکار شده می آید و بعد تعداد تقلب های آشکار شده می آید.

- خروجی درخواست wrist-up2 count هم شامل خط است که اول آیدی فردی که تقلب positive دارد می آید و سپس تعداد تقلب های آن شخص می آید.

- خروجی یدرخواست wrist-up3 count هم شامل خط است و در هر خط ابتدا آیدی امتحان و سپس تعداد

تقلب های positive شده در آن امتحان می آید.

- خروجی درخواست yyyy/mm/dd هم شامل count خط است که در هر خط ابتدا تاریخ روز به فرمت wrist-up4 سپس تعداد تقلب های positive شده در آن روز می آید.

- خروجی درخواست yyyy/mm/dd هم شامل count خط است و در هر خط ابتدا تاریخ روز به فرمت wrist-up5 سپس تعداد افراد مشکوک در آن روز می آید

## ورودی نمونه ۱

```
register asqar 1234 student
login asqar 1234
6
3
1 2
3 4
2 5
3
2
1 1 3 2021/01/20
1 3 1 2021/01/30
3
1 negative 1 2021/01/20
1 positive 2 2021/01/28
3 negative 1 2021/12/26
logout
register akbar 1234 teacher
login akbar 1234

wrist-up1 3
end
logout
exit
```

## خروجی نمونه ۱

```
register successful  
login successful  
logout successful  
register successful  
login successful  
wrist-up1  
1 1  
2 0  
3 0  
logout successful
```

توضیحاتی در مورد این نمونه در این نمونه فرد با id شماره یک در روز ۲۸ ام یک تلاش برای تقلب داشته است و همینطور در روز ۳۰ ام یک جا به جایی بین امتحانی داشته که در واقع از امتحان شماره ۳ به امتحان شماره یک رفته است. و چون در این مدت هنوز مشکوک بوده است، طبق توضیحات گفته شده، تقلب این شخص بر همه آشکار نمیشه و یک مورد تخلف برای او ثبت می شود. دقت کنید که فرد شماره ۲ هم با فرد شماره یک در یک گروه بوده است و این شخص شماره یک در جا او را لو می دهد ولی چون شخص شماره ۲ هیچ جایی ای ندادسته در نتیجه با اینکه مشوک بوده است ولی تقلب اون بر همه آشکار نمیشه و موردي براش ثبت نمیشه

## ورودی نمونه ۲

```
register asqar 1234 student  
login asqar 1234  
6  
3  
1 2  
3 4  
2 5  
3  
2  
1 1 3 2021/01/20  
1 3 1 2021/01/30  
5  
1 negative 1 2021/01/20  
1 positive 2 2021/01/28  
3 negative 1 2021/02/04
```

```
2 positive 3 2021/02/20
5 positive 1 2021/01/28
logout
register akbar 1234 teacher
login akbar 1234
wrist-up2 4
wrist-up3 2
wrist-up4 2 1
wrist-up5 3
data clear
logout
exit
```

## خروجی نمونه ۲

```
register successful
login successful
logout successful
register successful
login successful
wrist-up2
1 1
2 1
5 1
3 0
wrist-up3
1 1
2 1
wrist-up4
2021/01/28 1
wrist-up5
2021/01/28 3
2021/01/29 3
2021/01/30 3
logout successful
```

توضیحاتی در مورد این نمونه

در توضیح این نمونه هم می‌توان گفت که در درخواست wrist-up<sup>۲</sup> ما فقط افراد بیشترین تعداد تقلب positive رو می‌خواهیم که به وضوح، افراد با آیدی های یک و دو و پنج هر کدام  تقلب positive دارند و یغه هیچی در نتیجه خروجی ای بخش به شکل

#### wrist-up2

1	1
2	1
5	1
3	0

می‌شود

در درخواست wrist-up<sup>۳</sup> همانطور که مشاهده می‌شود، در در امتحان‌ها با آیدی های یک و دو سه هر کدام یک تقلب positive وجود دارد و در نتیجه خروجی این بخش به شکل

#### wrist-up3

1	1
1	2

خواهد بود چون ما تنها ۲ امتحان را خواسته ایم. در درخواست wrist-up<sup>۴</sup> چون در امتحان با آیدی شماره ۲ تنها یک تقلب positive و آن هم در روز ۲۸ ام رخ داده است، در نتیجه خروجی به شکل زیر می‌شود

#### wrist-up4

2021/01/28 1

می‌شود

و در نهایت خروجی درخواست wrist-up<sup>۵</sup> را به شدت توجه کنید.  
اول از همه فرد مشکوک، شخص با آیدی شماره یک است که در روز ۲۸ ام مشکوک شده است. حال دقت کنید که شخص شماره  ۲ با شخص شماره ۲ در یک گروه تقلب بوده اند و در نتیجه این فرد هم مشکوک می‌شود به تقلب و در عین حال شخص شماره ۵ هم در همین روز ۲۸ ام مش  وکبه تقلب شده است و چون هم گروهی

تقلب او شخص شماره ۲ است، تاثیر خاصی ندارد چون شخص شماره ۲ قبل تر مشکوک شده بود. پس تا اينجا در بازه‌ی ۲۸ام ژانویه تا پایان ۴ام فوریه، در هر روز، ۳نفر مشکوک بوده‌اند. حال در روز ۲۰ام فوریه، شخص شماره ۲ با موفقیت تقلبی انجام داده است و چون هم با فرد شماره ۱ و هم با فرد شماره ۵ در گروه هست، آن‌ها هم مشکوک می‌شوند و در نتیجه در بازه‌ی ۲۰ام فوریه تا ۲۷ام فوریه باز هم سه شخص ۱ و ۲ و ۵ مشکوک هستند پس در مجموع ۱۶روز وجود دارد که جمع مشکوک‌های آن برابر ۳است و چون ما تنها ۳روز را خواسته‌ایم در خروجی به ما نمایش بدهید و در صورت برابر بودن تعداد افراد دربرخی روز، روز قبل تر، بالاتر باید می‌آمد، خروجی در نهايیت به شکل

wrist-up5

2021/01/28 3

2021/01/29 3

2021/01/30 3

می‌شود