رفلکشن

سوال مربوط به مبحث Reflection است.

قرار است برنامه‌ای بنویسیم که مسئولیت ساخت نمونه‌ها را برعهده بگیرد.

ابتدا فایل‌های برنامه را از source دانلود نمایید.

متد init از کلاس Initializer را با توجه به توضیحات زیر پیاده‌سازی کنید.

متد init یک لیست از نام کلاس‌هایی که قرار است بر روی آن‌ها عملیاتی را انجام دهد، دریافت می‌کند. سپس از تمام کلاس‌های که دارای annotation از نوع Instantiate می‌باشند، یک شی می‌سازد. دقت کنید که باید از هر **نوع کلاس**، تنها یک بار نمونه ساخته شود. اگر در هنگام گرفتن شی Class یا ساخت کلاسی که دارای annotation مربوطه است، خطایی رخ داد، خطایی از نوع InitializeException پرتاب نمایید. (خطاهای دیگر مثل ClassNotFoundException را catch نموده و به جای آن خطای گفته شده را پرتاب نمایید)

بعد از ساخته‌شدن نمونه‌ها زمان آن می‌رسد که ارتباط بین نمونه‌ها فراهم شود.

در کلاس‌ها، فیلدهای مختلفی وجود دارد. برای **هر کلاسی که از آن نمونه ساخته‌ایم**، اگر فیلدی دارای annotation از نوع Connect است، باید مقدار آن برابر یک شی از کلاسی شود که نشان دهنده‌ی تایپ آن فیلد است و قبلا از آن نمونه ساخته‌ایم. اگر نمونه‌ای از آن ساخته نشده است یا آن کلاس وجود نداشت، خطایی از نوع InitializeException پرتاب نمایید. (خطاهای دیگر مثل ClassNotFoundException را catch نموده و به جای آن خطای گفته شده را پرتاب نمایید)

به نکات زیر برای حل سوال دقت کنید:

- برخی از فیلدهای کلاس ممکن است private باشد.
- برای حل سوال می‌توانید از متد getAnnotationType استفاده کنید.
- کالکشنی که توسط متد init برگردانده می‌شود شامل کلاس‌هایی است که از آن‌ها نمونه ساخته شده است. ما فقط نیاز به مجموعه‌ی کلاس‌ها داریم و نوع کالکشن خروجی مهم نیست.

تست

```
@Test
public void TestInitializer() {
    Collection<Object> objects = Initializer.init(
            Arrays.asList("user.Manager", "user.Service", "user.Developer"));

    assertEquals(3, objects.size());
    for (Object obj : objects) {
        if (obj instanceof Manager) {
            // there is a "Connect" annotation for Manager.developer
            assertNotNull(((Manager) obj).getDeveloper());
        } else if (obj instanceof Service) {
            // there is a "Connect" annotation for Service.manager
            assertNotNull(((Service) obj).getManager());
            // there is no "Connect" annotation for Service.employee
            assertNull(((Service) obj).getEmployee());
        } else if (obj instanceof Developer) {
            // an instance of Developer is created!

```

```
        } else {
            // anything else should fail
            fail();
        }
    }

    try {
        Initializer.init(Arrays.asList("user.Service", "user.Developer"));
        fail();
    } catch (Exception e) {
        // There is no instance of user.Manager
        // can't set manager for user.Service
        assertTrue(e instanceof InitializeException);
    }

    try {
        Initializer.init(Arrays.asList("user.notAvailable"));
        fail();
    } catch (Exception e) {
        // couldn't find class
        assertTrue(e instanceof InitializeException);
    }

    objects = Initializer.init(Arrays.asList("user.Developer", "user.Employee"));
    // user.Employee doesn't have Instantiate annotation
    assertEquals(1, objects.size());
    // objects contain only user.Developer
```

```
        assertEquals("user.Developer", objects.iterator().next().getClass().getName());
    }
```

آن چه که باید آپلود کنید

تنها فایلی که باید آپلود کنید یک فایل zip است که وقتی آن را باز می‌کنیم، در آن فقط یک فایل Initializer.java باشد.

چت کن

سوال مربوط به مبحث Thread و Serialization و Stream است.

پیشنهاد می‌شود حتما این تمرین را انجام دهید تا در پروژه‌ی پایانی با استفاده از ایده‌های این تمرین بهتر عمل کنید.
قرار است یک برنامه‌ی چت بنویسیم که در آن هر کسی می‌تواند به گروه کاربران آنلاین بپیوندد، هرزمان که خواست خارج شود و برای کاربر دلخواه پیام ارسال کند.
ابتدا فایل‌های برنامه را از source دانلود نمایید.

پیام‌های ارسال شده در این برنامه به یکی از صورت‌های زیر می‌باشند:

```
[name]:[message]
```

پیام message توسط کابر حاضر به کاربری با نام name ارسال می‌شود. اگر کاربری با آن نام آنلاین باشد، پیام به همین شکل برای آن کاربر ارسال خواهد شد که در آن به جای name، نام ارسال‌کننده‌ی پیام قرار می‌گیرد. در غیر این صورت، پیامی به شکل زیر برای خود کاربر حاضر ارسال می‌شود.

```
[name] doesn't exist
```

```
[name] connected
```

هرزمان که یک کاربر که به گروه چت می‌پیوندد، پیامی به شکل بالا برای تمام کاربران آنلاین چت توسط سرور ارسال می‌شود.

```
[name] disconnected
```

هرزمان که یک کاربر از گروه چت خارج شود، پیامی به شکل بالا برای تمام کاربران آنلاین چت توسط سرور ارسال می‌شود.

با توجه به توضیحات داده‌شده، برنامه را پیاده‌سازی کنید.

پروژه دارای سه package است که هر کدام در زیر توضیح داده شده‌اند.

بسته‌ی ViewModel

این بسته مربوط به مدل‌های استفاده شده در برنامه است و شامل ۳ فایل زیر می‌باشد:

- کلاس MessageType: این کلاس یک enum است که نوع پیام‌هایی را که بین سرور و کلاینت جابه‌جا می‌شود، مشخص می‌کند. این enum دارای ۴ مقدار زیر است:
  - نوع Connect: هنگام وصل شدن یک کلاینت به سرور، پیامی از این نوع برای سرور ارسال می‌شود. در صورت موفق بودن اتصال، سرور برای کابران آنلاین پیامی از همین نوع ارسال می‌کند.
  - نوع Disconnect: هنگامی که یک کلاینت چت را ترک می‌کند، پیامی از این نوع به سرور ارسال می‌نماید. سرور با دریافت پیام ارتباط را قطع‌کرده و در صورت احتیاج کارهای دیگری را هم انجام می‌دهد. (مثلا ارسال پیام غیرفعال شدن کلاینت به کاربران آنلاین چت)
  - نوع Text: هنگامی که یک کلاینت پیامی را به کلاینت دیگر را ارسال می‌کند، پیامی از این نوع از کلاینت به سرور ارسال شده و سرور آن پیام را به کلاینت مربوطه ارسال می‌کند.
  - نوع Error: هنگامی که یک کلاینت می‌خواهد پیامی را به کاربری ارسال کند که وجود ندارد یا آنلاین نیست، سرور پیامی از این نوع را برای او ارسال می‌کرد.

- کلاس Message: پیام‌های که میان سرور و کلاینت جابه‌جا می‌شود، از جنس این کلاس است (نه یک string ساده!!) این کلاس دارای ۴ ویژگی ارسال‌کننده‌ی پیام (sender)، دریافت‌کننده‌ی پیام (receiver)، متن پیام (messageText) و نوع پیام (messageType) است.

بسته‌ی Connection

این بسته برای ایجاد ارتباط بین کلاینت و سرور استفاده می‌شود و شامل ۳ کلاس می‌باشد:

- کلاس Connection: با اتصال هر کاربر به سرور، یک Connection با نام آن کاربر تولید می‌شود. (نام کاربران یکتا می‌باشد) هنگام ساخته شدن شی این کلاس، یک پیام از نوع Connect به سرور ارسال می‌شود. هر Connection دارای متدهای زیر است:
  - متد disconnect: ارتباط کاربر را با سرور قطع می‌نماید.
  - متد initializeServices: این متد برای بررسی برنامه است و بعد ساخت Connection فراخوانی می‌شود تا پیام‌ها توسط یک Thread پس از دریافت از سرور در خروجی نمایش داده شوند. (نمونه‌ی کد Chat را نگاه کنید)
  - متد sendText: این متد یک پیام، با متن مشخص (textMessage) را به یک کلاینت دیگر (receiver) ارسال می‌کند. تضمین می‌شود که پیامی از یک شخص به **خود شخص** ارسال نمی‌گردد.
- کلاس ListenerService: این کلاس برای آن است که به صورت concurrent پیام‌های ارسال‌شده از سرور به کلاینت را handle کند.
- کلاس ClientMessageHandler: این کلاس پیام‌های ارسال شده از سرور به کلاینت را handle می‌کند.

بسته‌ی Server

این بسته عملیات مربوط به سرور را handle می‌کند و شامل ۳ کلاس است:

- کلاس Server: این کلاس شامل متد start برای راه‌اندازی سرور می‌باشد. با اتصال هر کلاینت به سرور، یک ServerHandler به آن کلاینت اختصاص داده

می‌شود. تمام پیام‌های دریافت شده توسط سرور، به متد handle کلاس ServerHandler پاس داده خواهد شد.

- کلاس ServerHandler: برای کلاس ServerHandler متد handle این کلاس وظیفه دارد پیام‌های دریافت شده از کلاینت مربوط به خود را مدیریت کند. دقت کنید که سرور باید اطلاعات هر یک کلاینت وصل‌شده را ذخیره نماید تا بتواند با آن‌ها ارتباط برقرار نماید.
- کلاس User: این کلاس مدلی از یک کلاینت آنلاین است و ویژگی مهم آن username می‌باشد. شی‌های این کلاس توسط هیچ کلاینتی ساخته **نمی‌شوند** و فقط در اختیار **سرور** هستند. به عبارتی سرور پس از اتصال موفق کلاینت، شی مناسب از این جنس را ایجاد می‌کند و نگه می‌دارد.

آنچه که شما باید پیاده‌سازی کنید، کلاس‌های ClientMessageHandler و ServerHandler می‌باشند. توجه کنید که حق تغیر در هیچ‌یک از کلاس‌های دیگر را ندارید. فایل Chat جهت تست برنامه در حالت concurrent قرار داده شده است. شما باید کلاس Server را که دارای متد main است راه‌اندازی کرده و به تعداد دلخواه کلاینت توسط Chat ایجاد کنید.

مثال

آنچه که توسط کوئرا بررسی می‌شود فایل‌های مشابه ChatTest است که برنامه‌ی شما باید آن‌ها را پاس کند.

```
1  public class ChatTest{
2      @Test
3      public void test() throws InterruptedException {
4              Server.start();
5      Thread.sleep(100);
6      Connection Ali = new Connection("Ali");
7      Thread.sleep(200);
8      Connection Reza = new Connection("Reza");
9
10
```

```
Thread.sleep(200);
assertEquals("Reza connected", Ali.getRespond());
Thread.sleep(100);
Ali.sendText("Salam. Khobi?", "Reza");
Thread.sleep(100);
Reza.sendText("Keili kobam", "Ali");
Thread.sleep(100);
assertEquals("Reza:Keili kobam", Ali.getRespond());
Thread.sleep(100);
assertEquals("Ali:Salam. Khobi?", Reza.getRespond());
Thread.sleep(100);
Connection Amin = new Connection("Amin");
Thread.sleep(200);
assertEquals("Amin connected", Ali.getRespond());
Thread.sleep(100);
assertEquals("Amin connected", Reza.getRespond());
Thread.sleep(100);
Amin.sendText("Man ham omadam", "Ali");
Thread.sleep(100);
Amin.sendText("Man ham omadam", "Reza");
Thread.sleep(100);
assertEquals("Amin:Man ham omadam", Ali.getRespond());
Thread.sleep(100);
assertEquals("Amin:Man ham omadam", Reza.getRespond());
Thread.sleep(100);
Ali.disconnect();
Thread.sleep(200);
```

```
        assertEquals("Ali disconnected", Reza.getRespond());
        Thread.sleep(100);
        assertEquals("Ali disconnected", Amin.getRespond());
        Thread.sleep(100);
        Amin.sendText("Ali:koja rafti?", "Ali");
        Thread.sleep(100);
        assertEquals("Ali doesn't exist", Amin.getRespond());
        Thread.sleep(100);
        Amin.disconnect();
        Thread.sleep(200);
        assertEquals("Amin disconnected", Reza.getRespond());
        Thread.sleep(100);
        Reza.disconnect();
        Thread.sleep(200);
        }
}
```

آن چه که باید آپلود کنید

آنچه که باید آپلود کنید یک فایل zip است که وقتی آن را باز می‌کنیم، باید فقط شامل دو پوشه باشد. یک پوشه به نام Connection که در آن فقط کلاس ClientMessageHandler قرار دارد و پوشه‌ی دیگر به نام Server که در آن فقط یک کلاس ServerHandler قرار دارد.

گزارش آب و هوا

سوال مربوط به مبحث Java 8 Stream است.

اطلاعات مربوط به میزان بارش باران روزانه‌ی شهرهای مختلف در اختیار شما گذاشته می‌شود. شما باید گزارش‌های خواسته‌شده را در اختیار سازمان‌های هواشناسی قرار دهید.

ابتدا برنامه را از source دانلود نمایید و به آن نگاه بیاندازید.

در این برنامه اطلاعات هوا توسط کلاس Report دریافت می‌شود. این اطلاعات در قالب لیستی از Information ها می‌باشد و شما نباید آن را تغییر دهید. کلاس Information دارای ویژگی‌های زیر است:

- فیلد city: نام شهر
- فیلد date: نشان‌دهنده‌ی روزی است که در آن گزارش ثبت شده است. تاریخ در قالب YYYY/MM/DD است.
- فیلد amount: میزان بارندگی

اکنون شما بر اساس لیستی که به شما در کلاس Report داده می‌شود، باید دو متد زیر را پیاده‌سازی کنید:

- متد sumByCity(year): این متد یک لیستی از CityInformation برمی‌گرداند. برای هر شهر، مجموع میزان بارندگی را در آن سال برمی‌گرداند. اطلاعات خروجی بر اساس نام شهر مرتب شده‌اند.
- متد sumCityByMonth(year): این متد لیستی از CityMonthInformation برمی‌گرداند. برای هر شهر و هر ماه، مجموع میزان بارندگی را در آن سال برمی‌گرداند. اطلاعات خروجی ابتدا بر اساس نام شهر و سپس بر اساس ماه مرتب شده‌اند.

دقت کنید که در هر دوی این متدها لیستی که برگردانده می‌شود حاوی اطلاعاتی است که برای هواشناسی مفید باشد. یعنی **اگر در گزارشات، برای یک سال یا ماه، گزارشی وجود ندارد، نباید در گزارش‌های خروجی، مقدار صفر برای آن شهر یا ماه برگردانده شود.**

مثال

برای نمونه یک قسمت main در برنامه برای شما گذاشته شده است که خروجی آن باید به صورت زیر باشد:

```
sum total: 3
1: Hamadan 10
2: Kurdistan 52
2: Yazd 10
sum total: 3
1: Hamadan 5 10
2: Kurdistan 9 52
2: Yazd 2 10
```

آن چه که باید آپلود کنید

فایلی که باید آپلود کنید، یک فایل zip است که وقتی آن را باز می‌کنیم، در آن **فقط** کلاس Report.java قرار دارد.

مزایده

سوال مربوط به مبحث Thread است.

به مزایده خوش آمدید! قرار است به عنوان فروشنده، یک مزایده (Auction) را برگزار کرده و کالای خود را به بیشترین قیمت به فروش برسانید.
ابتدا کد برنامه را از source دانلود کنید.

لیست خریداران (Buyer) که قرار است به مزایده بیایند و قیمت پایه‌ای کالا (startPrice) از قبل مشخص است. شما منتظر می‌مانید تا تمامی خریداران در محل مزایده حضور پیدا کنند. بعد از حاضرشدن آن‌ها در محل مزایده، شما مزایده را آغاز می‌کنید. با شروع مزایده، به فاصله‌ی زمانی مشخصی (interval) منتظر دریافت یک پیشنهاد (Bid) از خریداران می‌مانید. به هنگام دریافت یک پیشنهاد دو حالت پیش می‌آید:

- خریدار در ابتدای مزایده، پیشنهادی **مساوی یا بیش‌تر** از startPrice می‌دهد یا پیشنهادی **بزرگ‌تر** از آخرین پیشنهاد (در صورت وجود) می‌دهد. در این حالت شما آن پیشنهاد را اعلام کرده و از نو، به مدت interval منتظر پیشنهاد جدید می‌مانید.
- در این مزایده افرادی هستند که قصد فریب دادن شما را دارند!!! این افراد در طول مزایده پیشنهادهای نامناسب یعنی کمتر از startPrice (در ابتدای مزایده) یا کمتر از بهترین پیشنهاد کنونی می‌دهند. در این حالت شما نه تنها به پیشنهاد توجه نمی‌کنید، بلکه به زمانی که باید صبر کنید **ادامه** می‌دهید. (از نو شروع به صبر کردن نمی‌کنید!!)

پیام‌هایی که در این برنامه چاپ می‌شوند شامل این موارد هستند:

به ازای هر بار که به مدت interval شروع به صبر کردن می‌کنید پیام زیر باید چاپ شود:

```
Auctioneer waiting for a bid
```

هربار که یک پیشنهاد را با توجه به توضیحات بالا قبول کردید، پیغام زیر را چاپ کنید. buyer نام خریدار و price قیمت پیشنهادشده از طرف اوست. (به فاصله‌ها دقت نمایید!)

```
Auctioneer receives a bid from [buyer] : [price]
```

اگر پس از تمام‌شدن interval در **ابتدای** مزایده، هیچ پیشنهاد مناسبی دریافت نکردید، پیام زیر چاپ خواهد شد. (این کار در متد startAuction انجام می‌شود و شما نباید آن را پیاده سازی کنید)

```
No good price suggested
```

اگر پس از مدت زمان interval (هنگامی که حداقل یک پیشنهاد مناسب دریافت کرده‌اید) دیگر پیشنهاد مناسبی دریافت نکردید، باید پیام زیر را چاپ کنید.

```
Auctioneer doesn't receive anything anymore
```

پس از چاپ این پیام خریداری که بهترین پیشنهاد را داده است، با پیامی به شکل زیر معرفی می‌شود. (این کار در متد startAuction انجام می‌شود و شما نیاز نیست آن را پیاده‌سازی کنید)

```
[buyer] give best price: [bestPrice]
```

توضیح کلاس‌ها

- کلاس Bid: این کلاس نشان‌دهنده‌ی یک پیشنهاد از سوی خریداران است. هر پیشنهاد شامل time و price است. time نشان دهنده‌ی زمان اعلام و price نشان دهنده‌ی قیمت آن پیشنهاد است.
- کلاس Buyer: این کلاس یک خریدار را نشان می‌دهد. هر خریدار دارای نام (name) و یک لیست از پیشنهادات است که در زمان مزایده اعلام می‌کند. در این صورت اگر لیست مزایده یک خریدار شامل Bid هایی باشد که time های آن‌ها به ترتیب ۲۰ و ۵۰ و ۸۰ است، یعنی شخص، اولین پیشنهاد خود را در زمان ۲۰، سپس مستقل از قبول‌شدن یا نشدن آن، دومین پیشنهاد را در زمان ۵۰ و به طور مشابه، سومین پیشنهاد را در زمان ۸۰ می‌دهد.
- کلاس Auctioneer: این کلاس نشان دهنده‌ی یک فروشنده است که دارای قیمت پایه (startPrice)، بازه زمانی (interval)، قیمت نهایی (finalPrice) و خریدار نهایی کالا (finalBuyer) می‌باشد.

به نکات زیر توجه کنید:

- آنچه که شما باید پیاده‌سازی کنید، متدهای run در دوکلاس Buyer و Auctioneer است. همچنین می‌توانید متد یا فیلدهای جدید هم به این دو کلاس اضافه کنید.
- حق تغییر در کلاس Bid را ندارید.
- زمان شروع مزایده را، **زمان صفر** در نظر بگیرید.
- اگر چه که معقولانه نیست، ولی واحد زمانی تمام زمان‌ها را برای فروشنده و خریداران، بر حسب **میلی‌ثانیه** در نظر بگیرید.

یک متد main در کلاس Auction برای شما قرار داده شده است تا با آن برنامه‌ی خود را تست نمایید. باید به ازای اجرای آن، برنامه‌ی شما خروجی زیر را تولید کند:

```
Auctioneer waiting for a bid
Auctioneer receives a bid from buyer2 : 400
Auctioneer waiting for a bid
Auctioneer receives a bid from buyer1 : 500
```

```
Auctioneer waiting for a bid
Auctioneer receives a bid from buyer1 : 700
Auctioneer waiting for a bid
Auctioneer receives a bid from buyer1 : 750
Auctioneer waiting for a bid
Auctioneer doesn't receive anything anymore
buyer1 give best price: 750
```

توضیحات خروجی:

فرض کنید می‌خواهید کالا حداقل ۲۰۰ ریال به فروش برسد. شما مزایده را آغاز می‌کنید. در ۵۰ میلی‌ثانیه‌ی اول مزایده، پیشنهاد به قیمت ۱۰۰ ریال داده می‌شود که شما آن را قبول نکرده و به اندازه‌ی ۴۵۰ میلی‌ثانیه باقی‌مانده، صبر می‌کنید. در میلی‌ثانیه‌ی ۲۰۰، دومین پیشنهاد با قیمت ۴۰۰ داده می‌شود. شما آن را قبول کرده و دوباره به مدت ۵۰۰ میلی‌ثانیه برای دریافت پیشنهاد بهتر صبر می‌کنید.

در میلی‌ثانیه‌ی ۴۰۰ (۲۰۰ میلی‌ثانیه پس از اولین پیشنهاد مناسب)، پیشنهاد سوم با قیمت ۵۰۰ داده می‌شود. با پذیرفته‌شدن آن، دوباره از نو، به مدت ۵۰۰ میلی‌ثانیه صبر می‌کنید.

در میلی‌ثانیه‌ی ۷۰۰ پیشنهادی با قیمت یک ریال داده می‌شود. (از همون پیشنهادهای فریب‌دهنده!!) شما به این پیشنهاد اهمیتی نداده و به اندازه‌ی ۲۰۰ میلی‌ثانیه‌ی باقی‌مانده صبر می‌کنید.

در میلی‌ثانیه‌ی ۸۵۰ (درست ۵۰ ثانیه قبل تمام‌شدن زمان انتظار)، پیشنهاد بهتری با قیمت ۷۰۰ ریال داده می‌شود. شما آن را قبول کرده و دوباره به مدت ۵۰۰ میلی‌ثانیه صبر می‌کنید.

نهایتا در میلی‌ثانیه‌ی ۱۳۰۰، آخرین پیشنهاد نیز داده می‌شود و شما آن را قبول کرده و به مدت ۵۰۰ میلی‌ثانیه‌ی دیگر صبر می‌کنید اما دیگر پیشنهادی اعلام نمی‌شود. پس شما پایان مزایده را اعلام نموده و کالا را به قیمت ۷۵۰ ریال می‌فروشید.

آن چه که باید آپلود کنید

فایلی که آپلود می‌کنید یک فایل zip است که وقتی آن را باز می‌کنیم در آن **فقط** دو کلاس Auctioneer و Buyer است.

بحرانی (امتیازی)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

جاسوسان برای شما نقشه‌ی شهرهای یک منطقه‌ی نظامی را فراهم نموده‌اند. در این نقشه n نقطه است که هر نقطه نماینده‌ی یک شهر است. همچنین بین شهرها m خط است. هر خط نشان‌دهنده‌ی این است که بین دو شهر، یک جاده‌ی **دوطرفه** است.

شما برای وارد کردن بیش‌ترین ضرر به دشمن، تصمیم می‌گیرید که تمام شهرهای حیاتی منطقه را با موشک بزنید.
شهر x بحرانی محسوب می‌شود اگر شهرهایی مانند y و z **وجود** داشته باشند که در صورت حذف شهر x، دیگر راهی برای رفتن از شهر y به z وجود نداشته باشد.

شما باید برنامه‌ای بنویسید که شهرهای بحرانی منطقه‌ی نظامی را مشخص کند.

ورودی

در ورودی دو عدد n و m که به ترتیب تعداد شهرها و تعداد راه‌های بین شهرهاست، داده می‌شود. سپس در m خط بعدی، هر خط دو عدد x و y آمده است که نشان‌دهنده‌ی این است که بین دو شهر x و y مسیری مستقیم وجود دارد. (ممکن است از یک شهر، به خود آن نیز مسیری باشد)

$$2 \leq n \leq 4000$$

$$1 \leq m \leq \binom{n}{2}$$

$$1 \leq x, y \leq n$$

خروجی

شهرهای بحرانی را به ترتیب صعودی چاپ کنید.

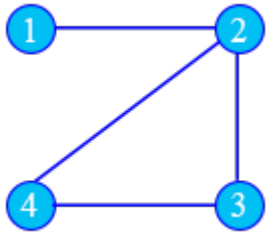
مثال

ورودی نمونه ۱

```
4 4
1 2
2 3
3 4
2 4
```

خروجی نمونه ۱

```
2
```

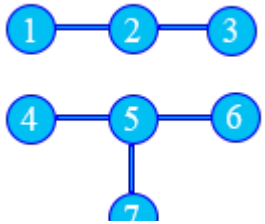


ورودی نمونه ۲

```
7 5
1 2
2 3
4 5
6 5
5 7
```

خروجی نمونه ۲

```
2 5
```

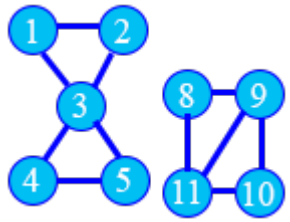


ورودی نمونه ۳

```
11 11
1 2
2 3
1 3
3 4
3 5
4 5
8 9
8 11
9 11
9 10
10 11
```

خروجی نمونه ۳

```
3
```




6
7