ویکی‌پدیا

- محدودیت زمان: ۰.۳ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

علیرضا فکر می‌کند دولت مخفی‌ای که کنترل جهان را به دست دارد، رمز عبور مغز کبوترها را لابه‌لای مقاله‌های ویکی‌پدیا قایم کرده است. برای همین قصد دارد متن مقاله‌ها را پردازش و این رمز را پیدا کند. اما از آنجایی که وقت به شدت تنگ است، تصمیم می‌گیرد تا از دوستانش کمک بگیرد و متن را به طور مساوی بینشان تقسیم کند تا همگی به طور موازی روی متن کار کنند.

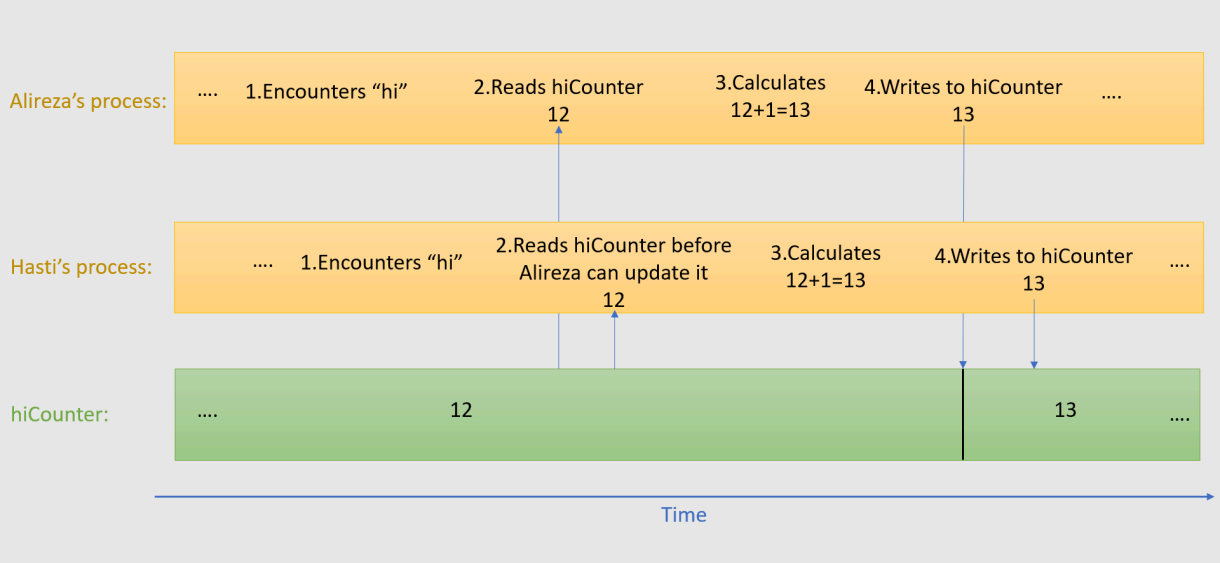
بخش اول

می‌خواهیم تعداد ظهور هر *کلمه* داخل متن ورودی را پیدا کنیم و داخل یک دفتر ثبت کنیم. برای این کار چندین روش وجود دارد:

روش ۱

▼ نمایش روش اول

همه‌ی افراد با یک دفتر مشترک کار کنند. یعنی هر کس به هر کلمه‌ای که رسید، دفتر را بردارد و داخلش شمارنده‌ی آن کلمه را یکی زیاد کند. حال فرض کنید علیرضا و دوستش هر دو (تقریباً) همزمان در متنشان به کلمه‌ی hi رسیده‌اند. اگر هر دو همزمان سعی کنند دفتر را بردارند و شمارنده‌ی hi را یکی زیاد کنند، ممکن است دچار خطا شوند (race condition):



پس در هر لحظه نباید بیش‌تر از یک نفر با دفتر کار کند. اما این محدودیت، سرعت را کم می‌کند.

روش ۲

▼ نمایش روش دوم

هر کس یک دفتر شخصی داشته باشد. پس از اتمام کار تمام افراد، نتایج دفترها را با هم ترکیب می‌کنیم تا به نتیجه‌ی نهایی برسیم. اما عمل ترکیب کردن ممکن است زمان زیادی طول بکشد.

روش ۳

▼ نمایش روش سوم

افراد با چند دفتر مشترک کار کنند. به این شکل که مجموعه‌ی تمام کلمات ممکن (U) را به k بخش افراز می‌کنیم:

$$A_1 \cup A_2 \cup ... \cup A_k = U$$

$$\forall i, j : A_i \cap A_j = \emptyset$$

و به هر بخش یک دفتر (که بین اعضای تیم مشترک است) اختصاص می‌دهیم. مثلا می‌گوییم تمام کلماتی که با حرف 'a' آغاز می‌شوند را در دفتر A می‌ریزیم، کلماتی که با 'b' آغاز می‌شوند را در دفتر B

می‌ریزیم، و به همین ترتیب. مثل روش ۱ در این روش هم دو نفر نباید همزمان روی یک دفتر کار کنند؛ اما برخلاف روش ۱ در حالتی که برای مثال علیرضا به کلمه‌ی hi و دوستش (در همان لحظه) به کلمه‌ی bye رسیده، دیگر نیازی نیست منتظر یکدیگر بمانند و می‌توانند همزمان یافته‌شان را ثبت کنند.

همچنین چون دفترها کلمه‌ی مشترکی ندارند، ترکیب نتیجه‌هایشان با هم به راحتی صورت می‌گیرد (با چسباندن دفترها به هم) و برخلاف روش دوم زمان‌بر نیست.

با این توصیفات شما باید از روش سوم استفاده کنید.

*تعریف *کلمه* در این سوال:** رشته‌ای از کاراکترهای الفبایی که بین کاراکترهای غیرالفبایی باشد (یا قبل/بعدش هیچ کاراکتری نباشد). الفبایی بودن یا نبودن یک کاراکتر را متد `Character.isAlphabetic` تعیین می‌کند.

بخش دوم

باید Hack Code متن ورودی را محاسبه کنید. این کد برای رشته‌ی S به طول k به شکل زیر محاسبه می‌شود:

$$s[k-1] * a^{k-1} + s[k-2] * a^{k-2} + ... + s[1] * a^1 + s[0] * a^0$$

که $s[0]$ راست‌ترین و $s[k-1]$ چپ‌ترین کاراکتر S و `a = 0.99998` است. کد را با دقت `double` محاسبه کنید. مثال:

$$hackCode(\text{"}bye\text{"}) = 98 * 0.99998^2 + 121 * 0.99998 + 101 \approx 319.9937$$

قالب

برای یکدستی میان دانشجویان، تابع main برنامه‌ی خود را در قالب زیر بنویسید:

```
public static void main(String[] args) {
    //Input:
    final String text;
```

```
    final int THREAD_COUNT;
    try {
        StringBuilder builder = new StringBuilder();
        BufferedReader bufferedReader = new BufferedReader(new Inpu
        THREAD_COUNT = Integer.parseInt(bufferedReader.readLine());
        String line;
        while (true) {
            line = bufferedReader.readLine();
            if (line.equals("!end"))
                break;
            builder.append(line).append("\n");
        }
        text = builder.substring(0, builder.length() - 1);
        bufferedReader.close();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    final long startTime = System.nanoTime();
    //Process:
    //Write your code here

    final long elapsedMillis = (System.nanoTime() - startTime) / 10
    //Output:
    //Print the results here
}
```

برای بررسی سرعت اجرای برنامه‌تان می‌توانید `elapsedMillis` را پرینت کنید (این برای کنجکاوی خودتان است و برای این سوال نیاز نیست). به دو نکته‌ی زیر توجه داشته باشید:

- برای ورودی‌های کوچک، multithreading ناکارآمد است و سرعت را افزایش نمی‌دهد.
- افزایش تعداد تردها لزوما به معنی افزایش سرعت نیست (چون با این کار تعداد تردهای سطح کاربر افزایش می‌یابد ولی همیشه تعداد تردهای CPU ثابت است. با این مورد در درس سیستم‌های عامل بیش‌تر آشنا می‌شوید).

راهنمایی

▼ نمایش راهنمایی‌ها

- با `synchronized` و `start()` و `join()` آشنا باشید.
- برای افراز کلمات به k گروه، می‌توانید از تابع `hashCode` استفاده کنید (نیاز به آشنایی با این تابع ندارید صرفاً بدانید یک رشته را تبدیل به یک int می‌کند). می‌توانید کلماتی که باقی‌مانده‌ی تقسیم hashCodeشان بر k برابر با `i` یا `-i` باشد را در گروه iام قرار دهید.
- در یک برنامه‌ی single-threaded، مقدار Hack Code می‌تواند به شکل زیر محاسبه شود:

```
double hackCode = 0;
for (int i = 0; i < S.length(); i++) {
  hackCode = hackCode * 0.99998 + S.charAt(i);
}
```

ورودی

در خط اول تعداد تردهایی که برنامه‌ی شما باید به صورت موازی اجرا کند می‌آید (تعداد دوستان علیرضا).

سپس متن اصلی داده می‌شود. خط آخر ورودی، با `end!` مشخص خواهد شد (این خط جزو متنی که باید پردازش شود **نیست**). تعداد کلمات متن حداکثر 15000 و طول هر کلمه حداکثر 45 است.

خروجی

ابتدا تعداد کلمات را خروجی دهید. سپس باید بلندترین کلمه و پرتکرارترین کلمه خروجی داده شوند (تضمین می‌شود این موارد یکتا هستند). برای محاسبه‌ی تعداد تکرار هر کلمه، کلمات را lowercase کنید. دقت کنید منظور ما از "کلمه" در سوال شرح داده شد. همچنین به فرمت خروجی در مثال‌ها توجه کنید.

در آخر باید Hack Code را با فرمت `%.3f` خروجی دهید.

سامانه‌ی کوئرا صحت خروجی کد شما را بررسی می‌کند. بهره‌گیری شما از multithreading در تحویل حضوری چک خواهد شد.

مثال

ورودی نمونه ۱

```
1
THIS is a veryVERYlong word
this is*not*[an] isogram: heLlo
    THIS is_not-a(single)word.
What's thiS?
!end
```

خروجی نمونه ۱

```
Word count: 20
The longest word is "veryVERYlong" with a length of 12.
The most frequent word is "this" with 4 appearances.
HackCode: 8885.749
```

ورودی نمونه ۲

این مثال از مقاله‌ی Big Tech ویکی‌پدیا است. آن را از این لینک ببینید.

خروجی نمونه ۲

```
Word count: 4848
The longest word is "financialization" with a length of 16.
The most frequent word is "the" with 283 appearances.
HackCode: 2258416.109
```

ورودی نمونه ۳

این مثال از مقاله‌ی Nineteen Eighty-Four ویکی‌پدیا است. آن را از این لینک ببینید.

خروجی نمونه ۳

```
Word count: 11402
The longest word is "enthusiastically" with a length of 16.
The most frequent word is "the" with 882 appearances.
HackCode: 3678095.388
```

برای مثال‌های بیش‌تر می‌توانید به مقاله‌های Apple Inc.، Economy of Iran و Frozen (2013 film) رجوع کنید.

چیزی که باید آپلود بکنید

فایل ارسالی .zip شما باید ویژگی های زیر را داشته باشد:

- برای ارسال پاسخ های خود، تمامِ محتویاتِ پوشه‌ی src خود را در یک فایل زیپ قرار دهید و آن فایل زیپ را ارسال کنید. برای مثال فایلِ زیپِ شما باید ساختاری مانند زیر داشته باشد:

```
├── Class1.java
├── Class2.java
├── Main.java
├── Package1
|   ├── Class3.java
|   └── Class4.java
└── Package2
    ├── Class5.java
    └── Class6.java
```

- همان گونه که در دیاگرام بالا نشان داده شده است، entry point شما باید نام Main.java را داشته باشد یعنی این فایل درون هیچ پوشه (package) دیگری نباشد.

حالت‌های مختلف نتیجه‌ی جاج به شرح زیر است:

```
❌ compileTest -> Your code does not compile (check imports?)
❌ judgeProject[i] | ❌ runtimeCheck[i] -> Your code throws runtime exc
```

```
❌ judgeProject[i] | ✅ runtimeCheck[i] -> Your code produces wrong ans
✅ judgeProject[i] | ✅ runtimeCheck[i] -> GG
```

دقت کنید فقط زمانی نمره‌ی یک تست را می‌گیرید که تست `judgeProject` PASS شود.

در صورتی که کد شما کلا کامپایل نمی‌شود به کمک دکمه‌ی اجرای تست نمونه می‌توانید دلیل آن را مشاهده کنید.

شبکه

در این سؤال شما باید یک کفش‌فروشی را پیاده‌سازی کنید. در این کفش‌فروشی سه نوع کفش با تعدادی موجودی در ابتدا وجود دارد. سپس هر مشتری با مقداری پول می‌تواند این کفش‌ها را بخرد. در ادامه، توضیحات و دستورات مورد نیاز آمده است.

توجه: در این سؤال شما باید یک شبکه به صورت Client-Server پیاده‌سازی کنید که کلاینت‌ها به سرور وصل می‌شوند و دستورات زیر در سرور هندل می‌شوند. توابع و ساختار کلی این سوال به صورت کد در اختیار شما قرار گرفته‌اند، شما باید این توابع را کامل یا در صورت نیاز توابع کمکی اضافه کنید.

دستورات

دستور register: دستور زیر باید مشتری با نام، آیدی و مقدار پول مشخص شده را به سرور اضافه کند:

```
register:id:name:money
```

در صورت وجود داشتن مشتری‌ای با این آیدی باید خطا برگردانده شود.

دستور login: دستور زیر باید مشتری با آیدی مشخص شده را لاگین کند:

```
login:id
```

در صورت وجود نداشتن مشتری‌ای با این آیدی باید خطا برگردانده شود.

دستور logout: دستور زیر باید مشتری لاگین شده را لاگ‌اوت کند:

```
logout
```

در صورتی که هیچ مشتری‌ای لاگین نباشد باید خطا برگردانده شود.

دستور get price: دستور زیر باید قیمت کالای با نام مشخص را برگرداند:

```
get price:shoe_name
```

در صورت وجود نداشتن کالایی با نام داده شده باید خطا برگردانده شود.

*دستور *get quantity:* دستور زیر باید تعداد کالای باقی مانده با نام مشخص شده را برگرداند:

```
get quantity:shoe_name
```

در صورت وجود داشتن کالایی با نام داده شده باید خطا برگردانده شود.

*دستور *get money:* دستور زیر باید مقدار پول باقی مانده شخص لاگین شده را برگرداند:

```
get money
```

در صورت لاگین نبودن هیچ مشتری‌ای باید خطا برگردانده شود.

*دستور *charge:* دستور زیر باید به میزان پول مشخص شده، پول شخص لاگین شده را زیاد کند:

```
charge:money
```

در صورت لاگین نبودن هیچ مشتری‌ای باید خطا برگردانده شود.

*دستور *purchase:* دستور زیر باید کالای مشخص شده را به میزان داده شده خریداری کند:

```
purchase:shoe_name:quantity
```

در صورت وجود نداشتن اسم کالا یا ناموجود بودن کالا یا نداشتن پول کافی باید خطا برگردانده شود.

نکات

1- توجه کنید که چند کاربر ممکن است همزمان یک کالا را سفارش دهند و درست بودن تعداد کالای باقی مانده را باید هندل کنید.

2- در صورت اشتباه بودن دستورات ورودی، مثلا عدد نبودن مقدار پول یا تعداد، یا کافی نبودن ورودی‌های هر دستور، باید خطای نادرست بودن ورودی داده شود.

3- توجه کنید در صورت لاگین نبودن هیچ مشتری‌ای، همچنان می‌شود قیمت کفش‌ها یا تعداد آنها را مشاهده کرد.

4- برای ارسال این تمرین، یک template در اختیار شما قرار داده شده است. سعی کنید تا حد امکان از این template پیروی کنید ولی اضافه یا کم کردن متدها تا وقتی که خواسته‌ی سؤال برآورده شود مشکلی ندارد. template را می‌توانید از این لینک دریافت کنید.

ترنسپایلر

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

در این سؤال شما باید یک ترنسپایلر ساده طراحی کنید. ترنسپایلرها ابزاری هستند که برای تبدیل زبان‌های برنامه‌نویسی به یکدیگر استفاده می‌شوند. ورودی ترنسپایلر کد نوشته شده به یک زبان برنامه‌نویسی و همچنین زبان خروجی می‌باشد و خروجی آن تبدیل شده همان کد به زبان خروجی است. شما باید برای این تمرین کدهای نوشته شده به ۳ زبان سی پلاس پلاس، جاوا و راست را به یکدیگر تبدیل کنید.

ابتدا فایل‌های سؤال را از این لینک دانلود کنید.

در پوشه Grammars می‌توانید ساختار دستورات هر زبان را ببینید. برای سادگی، از هر زبان عبارات بسیار کمی باید پیاده کنید (تنها `if` و `while` ، `else` و `switch`). برای آشنایی بیش‌تر با گرامر زبان‌ها و مفاهیمی همچون Abstract Syntex Tree و لکسر و پارسر می‌توانید از لینک ۱، لینک ۲، لینک ۳، لینک ۴ و لینک ۵ استفاده کنید.

در پوشه‌ی Languages، یک کلاس ابسترکت Code و یک اینترفیس Runnable مشاهده می‌کنید. در هر پوشه مربوط به هر زبان، یک کلاس با نام خود زبان مشاهده می‌کنید. شما باید تابع تبدیل AST به رشته کد را در هر کدام از زبان‌ها کامل کنید. **اکیدا توصیه می‌شود که فایل parser.cup هرکدام از زبان‌ها را بخوانید** تا با نحوه‌ی خروجی پارسر (Node مربوط به هر statement زبان در AST) آشنا شوید. همچنین دقت کنید که برای تبدیل درخت پارس‌ها به یکدیگر و ایجاد قطعه کد مربوطه، نکات زیر حائز اهمیت می‌باشد:

- در زبان سی پلاس پلاس می‌توان از عبارت `cout` استفاده کرد که ورودی‌های آن با `<<` از هم جدا می‌شوند و هر کدام در یک خط جدا پرینت می‌شود. (در واقعیت این عبارات پشت سر هم پرینت می‌شود ولی برای راحتی تمرین فرض می‌شود که هر کدام در خط جداگانه پرینت می‌شوند.) هنگام مقایسه `cout` و `print` زبان‌های دیگر باید این نکته را در نظر داشته باشید. برای مثال دو کد زیر باید **برابر در نظر گرفته شود**.

```
cout << x1 << x2 << x3;
```

```
System.out.println(x1);
System.out.println(x2);
System.out.println(x3);
```

- زبان‌های سی پلاس پلاس و جاوا می‌توانند در یک دستور چند متغیر تعریف و مقداردهی کنند ولی در راست باید هرکدام در یک دستور جدا تعریف و یا مقداردهی شوند که این دو مورد مشابه زیر باید **برابر در نظر گرفته شود**.

```
int x = 0, y = 1;
x = 2, y = x;
```

```
let x = 0;
let y = 1;
x = 2;
y = x;
```

- در زبان راست ساختار سوییچ کیس کمی با سی پلاس پلاس و جاوا تفاوت دارد. در این زبان هر کیس می‌تواند شامل چند عبارت شود و ساختاری مشابه زیر دارد:

```
match [variable] {
  [first_case_values] => [first_case_commands],
  [second_case_values] => [second_case_commands],
  .
  .
  _ => [default_commands]
}
```

با توجه به ساختار بالا، باید مقادیر مختلفی که با یک کیس مچ می‌شوند را به درستی با زبان‌های جاوا و سی پلاس پلاس مقایسه کنید. برای مثال دو کد زیر باید **برابر در نظر گرفته شود**.

```rust
match x {
  1 | 2 | 3 => println!("hello");   ,
  4 | 5 => println!("hello world"); ,
  _ => println("goodbye");
}
```

```cpp
switch (x) {
  case 1: cout << "hello";
  case 2: cout << "hello";
  case 3: cout << "hello";
  case 4: cout << "hello world";
  case 5: cout << "hello world";
  default: cout << "goodbye";
}
```

پس از نوشتن توابع `generateCode` مربوط به هر زبان، باید کلاس `Transpiler` را پر کنید. این کلاس یک کلاس جنریک است که در خود قطعه کدهایی به زبان‌های مختلف دارد. باید توابع قرار گرفته در `Transpiler` را با توجه به توضیحات و کامنت‌های گذاشته شده در کد پر کنید.

پس از اتمام کار، می‌توانید از فولدر Tests استفاده کنید تا پیاده‌سازی خود را تست کنید. در صورت صحت پیاده‌سازی خود، فایل‌های پروژه را به شکل zip ارسال کنید.