پروژه‌ی درس طراحی کامپایلرها ـ گروه ۲

# فاز ۱ پروژه: واژه‌یاب

آقای بهرامی

نسخه ۱

موعد تحویل: ۲۵ اردیبهشت ۹۹

# ۱ مقدمه

پیش از این که به سراغ مطالعه‌ی این مستند بروید، توصیه می‌کنیم که حتما مستند مربوط به معرفی زبان Decaf را مطالعه کنید.

در طول این پروژه که شامل ۳ فاز خواهد بود، شما به ترتیب قرار است واژه‌یاب[۱] ، ساختاریاب[۲] و نهایتا کدساز[۳] را پیاده‌سازی کنید. در واقع پس از انجام فاز آخر کامپایلر شما کامل می‌شود. همچنین این کار را باید با استفاده از یکی از ۴ روش زیر انجام دهید:

۱. استفاده از کتابخانه‌ی LARK[۴] در پایتون (دقت کنید که در صورت انتخاب این روش، تنها می‌توانید از حالت LALR(1) موجود در آن برای ساختاریاب استفاده کنید). این ابزار با گرفتن گرامر LALR(1) و کمی تنظیمات، کار کدساز و ساختاریاب را انجام می‌دهد و ساختاریاب را به کدساز وصل می‌کند.

۲. استفاده از Flex برای واژه‌یاب و سپس استفاده از Bison برای ساختاریاب و کدساز. در این حالت، ابتدا با استفاده از Flex واژه‌یاب را پیاده‌سازی می‌کنید و پس از آن گرامر LALR(1) را به Bison می‌دهید و آن را به واژه‌یابتان وصل می‌کنید. سپس کدساز و تحلیل‌گر معنایی را به کمک C++ پیاده‌سازی می‌کنید.

۳. استفاده از ابزار PGen برای تولید جدول پارس از روی گراف نحو و سپس استفاده از پایتون (یا جاوا) برای پیاده‌سازی باقی کامپایلر. ابزار PGen با گرفتن گراف نحو، درخت پارس مربوط به گرامر را خروجی می‌دهد. در این حالت برای واژه‌یاب می‌توانید از Regex برای این روش الگوریتم پارس به شما داده خواهد شد (ولی لزوما کار راحت‌تری در پیش نخواهید داشت).

در کلاس با بخش اول از روش دوم آشنا شدید. همچنین در کلاس‌های حل تمرین با روش اول بیشتر آشنا خواهید شد.

# ۲ کلیت این فاز

پس از این که ابزاری که می‌خواهید با آن پروژه را پیاده‌سازی کنید انتخاب کردید، باید به پیاده‌سازی واژه‌یاب بپردازید. در این بخش به عنوان ورودی یک stream از کاراکترها می‌گیرید و باید آن را به نشانه‌ها[۵] بشکنید. دقت کنید که هیچگونه عمل ساختاریابی‌ای در این بخش انجام نمی‌شود و هر نشانه مستقل از نشانه‌های قبلی (به صورت بدون حافظه) پیدا می‌شود. واژه‌یاب، فضاهای سفید[۶] که خارج از ثوابت رشته‌ای و کاراکتری‌اند را در نظر نمی‌گیرد و از آنها می‌گذرد. و باید کلیدواژه‌ها، ثوابت integer و double و رشته‌ای و کاراکتری و بولین، عملگرها و شناسه‌ها[۷] را تشخیص دهد. همچنین برای کلماتی که مشکلات واژه‌ای دارند خطا دهد.

---

[۱] scanner
[۲] parser
[۳] codegen
[۴] lark-parser
[۵] tokens
[۶] whitespaces
[۷] identifiers

# ۳ ورودی‌ها و خروجی‌ها

همانطور که گفته شد در ورودی دنباله‌ای از کاراکترها به شما داده می‌شود. برای مثال ورودی زیر را در نظر بگیرید:

```
class Program { void main () {} }
```

خروجی واژه‌یاب برای این برنامه مطابق زیر خواهد بود:

```
class
T_ID Program
{
void
T_ID main
(
)
{
}
}
```

حال به مثال کمی پیچیده‌تر زیر توجه کنید:

```
{-123-a35,id3a,+*;}[||===!=()&&]<><=>==
a[24]="7"; n!=if;
false,-if;true32;
forpar
```

همانطور که مشاهده می‌کنید در اینجا واژه‌یاب باید نشانه‌های به هم چسبیده‌ای را از هم تشخیص دهد. خروجی آن چیزی شبیه به این خواهد بود:

```
{
-
T_INTLITERAL 123
-
T_ID a35
,
T_ID id3a
,
+
*
;
}
[
||
==
=
!=
(
)
&&
]
<
>
```

```
<=
>=
=
T_ID a
[
T_INTLITERAL 24
]
=
T_STRINGLITERAL "7"
;
T_ID n
!=
if
;
T_BOOLEANLITERAL false
,
-
if
;
T_ID true32
;
T_ID forpar
```

دقت کنید که واژه‌یاب حتی اگر به یک نشانه رسیده باشد، تا جایی که با اضافه شدن کاراکتر بعدی به نشانه‌ی فعلی کماکان نشانه‌ای معتبر داشته باشیم، ادامه می‌دهد.

همچنین همانطور که در مثال‌ها مشخص است، برای Literal ها و شناسه‌ها، علاوه بر نوع نشانه‌ی تشخیص داده شده، باید خود نشانه را نیز ذکر کنید. مثلا برای شناسه‌ی true۳۲ در خروجی مقدار زیر چاپ شده:

```
T_ID true32
```

## ۱.۳ انواع نشانه‌ها

برای عمل‌گرها و کلیدواژه‌ها کافی‌است خودشان را خروجی دهید. برای شناسه‌ها، نوع نشانه T_ID است. برای ثوابت integer و double و رشته و boolean نیز نشانه‌ها از این قرار هستند:

```
integer: T_INTLITERAL
double: T_DOUBLELITERAL
string: T_STRINGLITERAL
bool: T_BOOLEANLITERAL
```

در هنگام چاپ کردن خروجی‌ها حتما به این نکته توجه کنید که در انتهای خطوط تنها یک \n چاپ کنید.

اگر در حین عملیات به جایی رسیدید که ابتدای stream باقیمانده را با هیچ token ای نمی‌شد مطابقت داد، در انتهای خروجی یک UNDEFINED_TOKEN بنویسید و برنامه را خاتمه دهید.

## ۴ چند نکته‌ی کلی

- برای هرکدام از روش‌های پیاده‌سازی، یک bash script به عنوان قالب به شما داده می‌شود. حتما دقت کنید که پروژه‌تان با استفاده از این script قابل اجرا روی هر ماشینی باشد. چرا که در هنگام تحویل، تنها از این طریق برنامه‌تان تست خواهد شد. بنابراین در استفاده از کتابخانه‌های غیر استاندارد دقت کنید و در صورت لزوم مسائل را با مسئول فنی پروژه مطرح کنید.
- دقت کنید که تمام بخش‌های پروژه باید توسط خود شما پیاده‌سازی شوند. در صورت کشف تقلب، نتنها نمره‌ای به شما تعلق نخواهد گرفت بلکه از نمره‌ی شما کسر خواهد شد.
- در صورتی که از منابعی استفاده می‌کنید، حتما آنها را ذکر کنید.
- در صورت مشاهده‌ی تقلب مطابق با سیاست‌های درس برخورد خواهد شد. که به این معنی است که نمره‌ی تمام مشارکت‌کنندگان (تقلب دهنده و تقلب گیرنده) از این فاز برابر با منفی حداکثر نمره‌ی قابل کسب از این فاز خواهد شد.
- سعی کنید پروژه را زودتر شروع کنید. با این که پروژه در سه فاز تقسیم شده، ولی اگر پیاده‌سازی را به روزهای نزدیک به ددلاین بیاندازید دچار مشکل خواهید شد. در روزهای ابتدایی برای انجام آن برنامه‌ریزی کنید.

## ۵ نکاتی در مورد تحویل

۱. به شما متناسب با روش‌هایی که برای پیاده‌سازی پروژه گفته شد، قالب‌هایی داده می‌شود. ساختار این قالب‌ها به این صورت است که شامل یک فولدر به نام tests هست که ورودی‌ها و خروجی‌های نمونه در آن قرار دارند و در نهایت هم تست‌کیس‌ها آنجا ریخته می‌شوند تا برنامه‌ی شما تست شود. همچنین یک bash script دارد به نام run.sh که از آن استفاده می‌شود تا برنامه‌های تست اجرا شوند. کد اصلی شما بسته به زبان انتخابی درون یکی از main.py/main.cpp/main.java قرار دارد که باید قابلیت گرفتن آرگومان ورودی و خروجی را داشته باشد (توسط i- و o-).

۲. در هر قالب، یک اسکریپت اجرایی وجود دارد که در نهایت برای تست کد شما تنها از همان استفاده خواهد شد.

۳. پیشنهاد می‌شود از همان ابتدای کار کد خود را طوری بزنید که با قالب مربوطه سازگاری داشته باشد تا بعداً دچار دردسر نشوید.

۴. اگر از package هایی استفاده می‌کنید، مطمئن شوید که آنها را به درستی وارد بخش پیش‌نیازهای قالب بکنید تا اجرای کدتان روی سیستم‌های دیگر دچار مشکل نشود.

۵. اگر از زبان پایتون استفاده می‌کنید، حداقل به نسخه‌ی ۳/۶ پایتون نیاز دارید. برای زبان جاوا به Oracle Java 8 و یا بالاتر نیاز دارید. همچنین اگر از زبان C++ استفاده می‌کنید، مطمین شوید که g++ 5.2 یا بالاتر را روی سیستمتان دارید.

۶. توجه کنید که کدهای فرستاده شده مثالی هستند برای نشان‌دادن نحوه‌ی کار با run.sh ولی به نحوه‌ی ورودی و خروجی دادن به آنها دقت کنید:

- Python

```
python3 main.py -i <input file> -o <output file>
```

- C++ و Java

اگر اسم فایل خروجی main باشد:

```
main -i <input> -o <output>
```

۷. توجه کنید که خروجی تست‌ها در پوشه‌ی out ریخته می‌شود.

۸. در نهایت نمره‌ی شما از این بخش برابر با تعداد تست‌هایی می‌شود که توسط کد شما پاس شوند.

۹. محل آپلود پروژه روی سایت کوئرا است.

۱۰. پس از گذشتن موعد تحویل، تا ۴ روز می‌توانید کدتان را آپلود کنید که به ترتیب منجر به جریمه‌ی ۱۰ درصدی، جریمه‌ی ۱۵ درصدی، جریمه‌ی ۴۰ درصدی و جریمه‌ی ۵۰ درصدی می‌شوند.

## ۶ پرسش و پاسخ

گروه حل تمرین آماده پاسخگویی به شما در طول مدت پروژه خواهند بود. با این حال لطفاً سوالات خود را برای روزهای پایانی نگذارید، زیرا ممکن است زمان پاسخ افزایش یابد.

- در صورتی که به ابهامی در مورد زبان Decaf یا تعریف فازهای پروژه برخوردید با اشکان میرزایی از طریق ایمیل مطرح کنید:

  amirzaei@ce.sharif.edu

- مشکلات مربوط به مسائل فنی را از بردیا ابهری بپرسید:

  bardia.abhari13@gmail.com

- در صورتی که در مورد LARK سوالی داشتید از علیرضا طباطبائیان بپرسید:

  tabanavid77@gmail.com

- در صورتی که با تست‌های ارائه شده مشکل دارید، با سعید اکبری در میان بگذارید:

  xsaeidscorp@gmail.com

## ۷ گروه‌بندی

تا ۱۷ اردیبهشت گروه‌هایتان را در گوگل فرم زیر وارد کنید. دقت کنید که این زمان به هیچ وجه تمدید نخواهد شد. گروه‌ها می‌توانند ۲ تا ۳ نفره باشند. همچنین اگر اسم کسی در بیش از یک گروه باشد، همه‌ی گروه‌ها به محض مشاهده حذف خواهند شد.

برای دیدن فرم، اینجا را کلیک کنید.