پروژه‌ی درس طراحی کامپایلر

فاز ۲ پروژه: ساختاریاب

نسخه ۱.۱

موعد تحویل: ۵ آذر

۱ مقدمه

در این فاز پروژه قرار است ساختاریاب را پیاده‌سازی کنید و به واژه‌یابی که در فاز قبلی پروژه پیاده‌سازی کرده‌اید متصلش کنید. توجه کنید که پس از پیاده‌سازی این بخش از آنجایی که هنوز به کدساز نرسیده‌ایم، نمی‌توان کدی را اجرا کرد و از صحت آن مطمئن شد. به همین دلیل تنها از شما خواسته می‌شود که با گرفتن یک کد ورودی به زبان Decaf تعیین کنید که آیا خطای نحوی [۱] دارد یا خیر.

۲ انواع خطاهای زمان کامپایل

گفتیم که باید در این بخش مشخص کنید که برنامه خطای نحوی دارد یا نه. یعنی چه؟ بگذارید نگاهی به انواع خطاهای زمان کامپایل بیاندازیم:

- خطای واژه‌ای [۲]

  شامل غلط‌های املایی در نام شناسه‌ها، کلیدواژه‌ها و عملگرها است. این خطاها عموما توسط واژه‌یاب کشف می‌شوند.

- خطاهای نحوی

  خطاهایی از قبیل جا افتادن ؛ و یا درست نبودن پرانتزگذاری و ... که در واقع رعایت نکردن syntax برنامه‌اند از این قبیل خطاها اند. این گونه خطاها عموما توسط ساختاریاب کشف می‌شود.

- خطاهای معنایی [۳]

  خطاهایی نظیر استفاده از متغیر تعریف نشده یا عدم همخوانی نوع در هنگام انتساب و ... از این دست خطاها هستند. این موارد عموما توسط کدساز کشف می‌شوند.

- خطاهای منطقی [۴]

  به این معنی است که منطقی که برنامه انجام می‌دهد با چیزی که برنامه‌نویس انتظارش را داشته متفاوت است. ولی برنامه درست کامپایل می‌شود و اجرا می‌شود. همچنین حلقه‌های نامتناهی و کدهای غیرقابل دسترس از این دست خطاها هستند. اکثر خطاهای منطقی به هیچ عنوان قابل کشف نیستند.

منبع این بخش اینجا است. می‌توانید برای مطالعه‌ی بیشتر به آن مراجعه کنید.

۳ خطای‌های معنایی زمان کامپایل در Decaf

۱.۳ عملیات با انواع غیر هم‌خوان

همان‌طور که در مستند توضیحات زبان آمده، عملیات‌ها محدودیت‌هایی روی نوع‌های عملگرهایشان دارند و هم‌خوانی عملگرها باید در آن‌ها رعایت شود. به مثال‌های زیر توجه کنید.

---

[۱] syntax-error
[۲] lexical-error
[۳] semantical-error
[۴] logical-error

```
6.7 + 7 //float + int
"hello" + 7 //string + int
```

۲.۳ انتساب انواع غیر هم‌خوان

در انتساب باید هم‌خوانی میان اجزا رعایت شود، به مثال‌های زیر توجه کنید.

```
int a;
a = 1.5
----
class1 A; //class is not in ancestors of class2
class2 B;
A = B;
```

۳.۳ استفاده از متغیری که تعریف نشده است یا دسترسی به آن امکان‌پذیر نیست

در صورتی که در یک حوزه [۵] و یا حوزه‌های بالاتر یک متغیر تعریف نشده باشد در صورت استفاده از آن متغیر، خطا رخ می‌دهد. همچنین در صورتی که قواعد دسترسی به متغیرها رعایت نشده باشد، خطا خواهیم داشت.

۴.۳ صدا زدن تابعی که موجود نیست یا دسترسی به آن امکان‌پذیر نیست

در صورتی که یک تابع موجود نباشد و یا براساس قواعد دسترسی نتوان آن را صدا زد، در صورت فراخوانی با خطا مواجه می‌شویم.

۵.۳ ناهم‌خوانی نوع خروجی تابع با نوع مورد انتظار آن

نوعی که درون توابع بازمی‌گردانیم، باید با نوع خروجی تعریف شده در امضای [۶] آن هم‌خوانی داشته باشد.

```
string f() {
   return 10;
}
```

دقت کنید مثال زیر خطایی ندارد.

```
Parent f() {
   Child ch;
   return ch;
} // Parent is in ancestors of Child
```

۶.۳ ناهم‌خوانی پارامترهای واقعی تابع حین فراخوانی با پارامترهای رسمی آن

پارامترهایی که تابع را با آن صدا می‌کنیم باید با پارامترهای تعریف شده در امضای آن هم‌خوانی داشته باشد.

---

[۵] scope
[۶] signature

```
int f(int a, int b) {
 ....
}
f(1.5, 2);
```

دقت کنید مثال زیر خطایی ندارد.

```
int f(Parent p) {
 ....
}
Child ch;
f(ch);
// Parent is in ancestors of Child
```

۷.۳ استفاده از نوع غیرصحیح در اندیس‌دهی آرایه یا ساخت آن (طول آرایه)

در اندیس آرایه، حتما باید از نوع int استفاده کنیم. هم‌چنین طول آرایه نیز باید int باشد.

منظور از نوع در خطاهای بالا، نوع ایستای [۷] متغیرهاست.

در صورت مشاهده هر یک از خطاهای فوق، عبارت Semantic Error را چاپ کنید و به برنامه خاتمه دهید.

۴ ورودی و خروجی

برنامه‌ی شما در ورودی چیزی را دریافت می‌کند که انتظار می‌رود یک کد به زبان Decaf باشد. ولی ممکن است یک دنباله‌ی کاملا بی‌معنی از کاراکترها باشد. در خروجی، اگر کد ورودی دارای خطای نحوی بود، عبارت Syntax Error و در غیر این صورت اگر به هر طریقی با استفاده از گرامر داده شده در مستند توصیف زبان Decaf قابل تولید بود عبارت OK را چاپ کنید.

برای مثلا به برنامه‌های زیر توجه کنید:

---

[۷] static

```
public class Main { // error 1: program should start with 'class', not
    identifier 'public'
   static void main() {
      int x = 1;
      int y;
      y = 1;
      z[] = 2; // error 2: missing index for '[]'
   }
}
```

همانطور که در کامنت‌ها مشاهده می‌کنید، این برنامه دارای دو خطای نحوی می‌باشد. پس در خروجی باید Syntax Error چاپ شود. کد زیر نیز خطای نحوی دارد و باید پیغام Syntax Error در خروجی چاپ شود:

```
class Main {
   int x,y; // error 1: this grammar is not supported

   static void main() {
      x = 0xf;
      y = 0xff;
      ;
      Print(z,); // error 2: missing parameter after ','
      return 0;
   }
}
```

۵ نکاتی در مورد تحویل

در این فاز، آنچه که از شما تحویل گرفته می‌شود، صرفا بررسی خطاهای نحوی کد ورودی است. در حقیقت در تحویل این فاز شما نباید خطاهای معنایی را بررسی کنید و خروجی شما باید در صورتی که کد ورودی در گرامر پذیرفته می‌شود OK و در غیر این‌صورت Syntax Error باشد. اما در فاز سوم قسمتی از تست‌ها مربوط به کدهایی است که فاقد اشکال نحوی هستند اما اشکال معنایی دارند و برنامه شما باید خروجی Semantic Error را به ازای آن تست‌ها چاپ کند.
توصیه ما به شما این است که در این فاز، علاوه بر پیاده‌سازی بررسی نحوی، تشخیص خطاهای معنایی را نیز پیاده‌سازی کنید. در طول ترم به تدریج با مباحث لازم برای پیاده‌سازی بررسی معنایی آشنا خواهید شد لذا سعی کنید به تدریج این امکانات را نیز به برنامه‌تان اضافه کنید تا در فاز سوم صرفا به پیاده‌سازی کدساز بپردازید و دید بهتری برای فاز نهایی داشته باشید.

۶ چند نکته

- 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.
- دقت کنید که تمام بخش‌های پروژه باید توسط خود شما پیاده‌سازی شوند.
- در صورتی که از منابعی استفاده می‌کنید، حتما آنها را ذکر کنید.

- در صورت مشاهده‌ی تقلب مطابق با سیاست‌های درس برخورد 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.

گفتم: چشمم، گفت: به راهش می‌دار گفتم: جگرم، گفت پر آهش می‌دار
گفتم که: دلم، گفت: چه داری در دل گفتم: غم تو، گفت: نگاهش می‌دار