به نام خدا

فاز دوم پروژه کامپایلرها و زبان های برنامه‌نویسی

پاییز ۹۸

مهلت تحویل: ۲۷ آبان

در فاز قبل، تحلیل‌گر لغوی و نحوی را برای زبان acton پیاده‌سازی کردید. در این فاز شما باید اطلاعات مربوط به actor ها، msghandler های هریک از آن‌ها، scope های برنامه و متغیر های آن‌ها را جمع‌آوری کرده و در جدول علائم[1] ذخیره کنید. برای این کار شما باید درخت AST[2] مربوط به برنامه را در این فاز تشکیل دهید.

به منظور سادگی کار شما، یک شمای کلی از جدول علائم در اختیارتان قرار گرفته که بایستی آن را به نحو مناسب استفاده و پر نمایید. همچنین نودهای درخت AST در اختیار شما قرار گرفته که باید با توجه به آن‌ها با تعریف action های مناسب در گرامرتان، درخت را بسازید.

در این فاز، به هر متغیری که در کد تعریف شده، یک index نسبت می‌دهید. index ها از صفر شروع شده و به صورت auto increment (یا هر روش دیگری به شرطی که مقدار آن‌ها برای هر متغیر یکتا باشد) به متغیر ها assign می‌شوند.

در این فاز باید موارد زیر را بررسی کرده و در صورت خطا، پیام مناسب را به فرمتی که برای هرخطا گفته شده به ترتیب شماره ی خط پرینت کنید و به بررسی کد تا انتهای فایل ادامه دهید. فرمت کلی هر یک از خطاهایی که باید نمایش دهید به شکل زیر است:

Line:<LineNumber>:<ErrorItemMessage>

مثال:

Line:21:Redifinition of actor A

که ErrorItemMessage برای هر یک از موارد زیر تعریف می‌شود.

---

[1] Symbol Table
[2] Abstract Syntax Tree

۱. عدم وجود دو actor با نام یکسان در برنامه

در صورت خطا در این قسمت باید خطای مربوطه اعلام گردد و یک نام موقت برای اکتور کنونی در نظر گرفته شود. برای اسم دادن دوباره به اکتور، از الگویی استفاده کنید که برخوردی با نام اکتور از پیش موجود در برنامه نداشته باشد.

ErrorItemMessage: Redifinition of actor <ActorName>

۲. عدم وجود دو متغیر با نام یکسان در یک scope

درصورت خطا همانند بخش قبل عمل کنید. دقت شود که با توجه به اینکه actorvar overloading نداریم، این خطا شامل actorvar ها و knownactor های اکتوری که از اکتور دیگر ارث‌بری می‌کند نیز می‌شود.

ErrorItemMessage: Redefinition of variable <VariableName>

۳. عدم تعریف دو msghandler هم‌نام در یک actor

درصورت خطا همانند بخش قبل عمل کنید. دقت شود که با توجه به اینکه msghandler overriding نداریم، این خطا شامل msghandler هایی که یک اکتور از آن‌ها ارث‌بری می‌کند نیز می‌شود. initial از این مورد مستثنی است.

ErrorItemMessage: Redefinition of msghandler <MsgHandlerName>

۴. عدم تعریف آرایه با طول مساوی صفر

ErrorItemMessage: Array size must be positive

۵. عدم تعریف اکتور با طول صف مساوی صفر

ErrorItemMessage: Queue size must be positive

۶. وراثت حلقوی

ErrorItemMessage: Cyclic inheritance involving actor <ActorName>

توجه کنید که این خطا تنها باید از اولین اکتور داخل یک حلقه ی وراثت گرفته شود.

پس از پیمایش کامل کد در حالتی که ورودی هیچ خطایی نداشته باشد، شما باید پیمایش preorder درخت AST را در خروجی چاپ کنید. (به ازای هر نود، خروجی تابع toString آن را در یک خط خروجی چاپ کنید.)

نکات مهم:

- گرامر زبان در اختیارتان قرار داده شده است. می‌توانید از آن یا گرامری که خودتان نوشتید استفاده کنید. استفاده از گرامر خودتان اندکی نمره امتیازی خواهد داشت اما توصیه می‌شود در صورت عدم اطمینان از آن از گرامری که در اختیارتان قرار می‌گیرد استفاده کنید. اگر از گرامر خودتان استفاده می‌کنید، قاعده ی شروع گرامر باید program باشد.
- با توجه به اینکه خروجی های شما به صورت خودکار تست می‌شوند، لازم است تا به این نکات در هنگام آپلود توجه کنید: یک فایل با فرمت stdID1_stdID2.zip (اگر گروهی تک نفره هستید stdID.zip) آپلود کنید که در آن فایل های شما قرار دارد. فولدر main که شامل کدهای AST، symbol table، visitor و فایل اصلی Acton.java میباشد در اختیار شما قرار داده شده است و باید مستقیماً در سورس پروژه قرار گیرد.
- در هنگام اجرای تست کیس‌ها، فایل Acton.java اجرا خواهد شد. در این فایل پس از فراخوانی قانون اول گرامر (که پارسر زبان Acton را اجرا خواهد کرد و درخت ast در حین اجرای آن ساخته خواهد شد)، باید کد مربوط به پیمایش ast را قرار دهید. در نهایت کافیست علاوه بر کامپایل کردن فایل های جاوا و گرامرتان و این فایل همانند قبل، این فایل را با یک آرگومان (اسم فایل تستتان، مثلاً test.act) اجرا کنید.