پروژه درس کامپایلر — نسخه فشرده (MiniLang)

نکات مهم

- پروژه در قالب گروه‌های دو یا سه‌نفره قابل انجام است. هر چه تعداد اعضای گروه بیشتر باشد، سخت‌گیری بیشتری در ارزیابی بخش‌های پروژه خواهیم داشت و نمره‌ی هر فرد نیز با توجه به میزان تسلط او در ارائه و مستقل از سایر اعضا تعیین می‌شود.
- برای تحلیل لغوی و نحوی خودکار، از ابزارهای مولد scanner و parser استفاده کنید: Flex/Bison (زبان C)، JFlex/CUP (زبان Java)، PLY (زبان Python)، Alex/Happy (زبان Haskell) یا معادل‌های دیگر. توجه: در پیاده‌سازی دستی فاز اول نیازی به این ابزارها نیست.
- زبان برنامه‌نویسی پیاده‌سازی پروژه آزاد است (C/C++, Java, Python, .NET و غیره).
- کامپایلر باید برنامه‌ی ورودی را تحلیل لغوی، سپس تحلیل نحوی و معنایی کرده و در نهایت کد میانی (سه‌آدرسه، مطابق آنچه در کلاس استفاده شده) تولید کند.
- در هر فاز باید گزارش مکتوب از برنامه‌های ورودی تست‌شده، خروجی آن‌ها و مستندات بخش‌های دستی (مانند NFA/DFA) ارائه شود.
- یک رابط کاربری ساده (حتی در سطح خط‌فرمان یا چاپ در فایل خروجی) برای نمایش خروجی هر فاز کافی است؛ طراحی رابط گرافیکی پیچیده الزامی نیست.

توصیه‌ها

1. کار را زود شروع کنید؛ یادگیری ابزارهای ناآشنا مانند مولدهای parser و scanner می‌تواند زمان زیادی ببرد.
2. هر فاز بر مبنای فاز قبلی است؛ اگر بخشی را ناقص رها کنید، در فازهای بعدی با مشکل مواجه خواهید شد.
3. در صورت نیاز می‌توانید گرامر MiniLang را در جزئیات (نه ساختار کلی) متناسب با ابزار انتخابی تغییر دهید.
4. از همان ابتدا، چند برنامه‌ی نمونه‌ی ورودی (درست و دارای خطا) برای هر فاز آماده کنید تا تست‌کردن کامپایلر در طول پروژه ساده‌تر شود.

نمره‌دهی

جدول زیر سهم نمره‌ی هر فاز از نمره‌ی کل پروژه (۵ نمره) را نشان می‌دهد.

نمره	ابزار/روش	فاز
۰.۵	NFA → DFA → جدول	اول: تحلیل لغوی به‌صورت دستی
۰.۵	Flex / PLY / JFlex / Alex	دوم: تحلیل لغوی با ابزار خودکار
۲	Yacc / Bison / CUP / Happy	سوم: تحلیل نحوی + معنایی پایه
۲	کد سه‌آدرسه	چهارم: تولید کد میانی
۵		**جمع کل**

زبان MiniLang

برنامه‌ی MiniLang از یک دنباله از دستورات و تعریف متغیرها در یک بلوک سراسری تشکیل می‌شود (بدون نیاز به تابع main یا توابع کاربر).

ملاحظات لغوی

- تمام کلمات کلیدی به‌صورت lowercase نوشته می‌شوند و زبان حساس به بزرگی و کوچکی حروف (case-sensitive) است.
- توضیحات (Comments): کامنت تک‌خطی با // و کامنت چندخطی با /* ... */؛ کامنت‌ها در تحلیل لغوی نادیده گرفته می‌شوند.
- فاصله‌ها (Whitespace): space، tab و line break مجاز و در تحلیل لغوی حذف می‌شوند ولی برای جداسازی توکن‌ها ضروری‌اند.
- شناسه‌ها (Identifiers): با حرف یا underscore شروع می‌شوند و می‌توانند شامل حروف، رقم یا underscore باشند (مثال: indexValue، total1، _sum، a).
- انواع داده: فقط int، bool و string.

- اعداد صحیح (int literal): دنباله‌ای از رقم‌های [0-9]+.
- رشته‌ها (string literal): بین دو علامت نقل‌قول، مانند "Hello"؛ پشتیبانی از escape sequence مانند \n و \t الزامی نیست (می‌تواند امتیاز اختیاری باشد).
- مقادیر منطقی: true و false.

کلمات کلیدی (Reserved Words)
if else while int bool string print input true false

عملگرها و جداکننده‌ها

عملگرهای محاسباتی: + - * /

عملگرهای رابطه‌ای و تساوی: < > == !=

عملگرهای منطقی: && || !

عملگر انتساب: =

جداکننده‌ها: ( ) { } ; ,

گرامر مرجع MiniLang

نمادهای گرامری مطابق زیر است:

<foo> غیرپایانه، foo پایانه/توکن، [x] به معنی صفر یا یک x (اختیاری)، x* صفر یا چند x، x+ یک یا چند x، | جداکننده‌ی جایگزین‌ها، و {} برای گروه‌بندی.

```
<program>     → <stmt_list>
<stmt_list>   → <stmt_list> <stmt> | ε
<stmt>        → <var_decl>
              | <assignment> ;
              | <if_stmt>
              | <while_stmt>
              | <io_stmt> ;
              | <block>

<var_decl>    → <type> <id> ;
```

```
<type>        → int | bool | string

<assignment>  → <id> = <expr>

<if_stmt>     → if ( <expr> ) <block> [ else <block> ]
<while_stmt>  → while ( <expr> ) <block>
<io_stmt>     → print ( <expr> ) | input ( <id> )
<block>       → { <stmt_list> }

<expr>        → <expr> <bin_op> <expr>
              | ( <expr> )
              | <unary_op> <expr>
              | <literal>
              | <id>

<bin_op>      → <arith_op> | <rel_op> | <eq_op> | <cond_op>
<arith_op>    → + | - | * | /
<rel_op>      → < | >
<eq_op>       → == | !=
<cond_op>     → && | ||
<unary_op>    → - | !

<literal>     → <int_literal> | <bool_literal> | <string_literal>
<id>          → <alpha> <alpha_num>*
<alpha_num>   → <alpha> | <digit>
<alpha>       → a | b | ... | z | A | B | ... | Z | _
<digit>       → 0 | 1 | ... | 9
<int_literal>  → <digit>+
<bool_literal> → true | false
<string_literal> → " <char>* "
```

فاز اول: تحلیل لغوی به‌صورت دستی

در این فاز لازم است تحلیل‌گر لغوی را به‌طور کامل دستی (بدون استفاده از ابزارهای مولد) پیاده‌سازی کنید. با توجه به حجم کار دستی، این بخش تنها برای زیرمجموعه‌ی زیر از زبان پیاده‌سازی می‌شود:

- شناسه‌ها (identifier)
- اعداد صحیح (int literal)
- عملگرهای + و = و ==
- پرانتزهای باز و بسته ( )
- دو کلمه‌ی کلیدی: if و while

مراحل پیاده‌سازی:

5. از روی این زیرمجموعه، چند نمونه‌ی ساختار لغوی استخراج کرده و عبارت منظم (Regular Expression) متناظر آن‌ها را بنویسید.
6. معادل NFA هر عبارت منظم را (روی کاغذ یا با نرم‌افزار) ترسیم کنید.
7. NFA به‌دست‌آمده را به DFA تبدیل کنید.
8. DFA را به جدول انتقال (مشابه اسلایدهای درس) تبدیل کنید.
9. با استفاده از جدول به‌دست‌آمده، کد تحلیل‌گر لغوی را بنویسید به‌طوری که با دریافت یک برنامه‌ی ورودی، مجموعه‌ای از TOKENها (شامل رشته‌ی تشکیل‌دهنده و کلاس/نقش آن) را به‌عنوان خروجی تولید کند. همچنین باید کاراکترهای غیرمجاز را تشخیص داده، پیغام خطای مناسب نمایش دهد و اسکن را پس از خطا ادامه دهد.

فاز دوم: تحلیل لغوی با ابزار خودکار

در این فاز با استفاده از ابزارهای مولد تحلیل‌گر لغوی (مانند PLY، JFlex، Flex یا Alex)، تحلیل‌گر لغوی را برای کل زبان MiniLang (نه فقط زیرمجموعه‌ی فاز اول) طراحی کنید. عبارات منظم متناظر با تمام توکن‌های گرامر MiniLang (کلمات کلیدی، شناسه‌ها، اعداد، رشته‌ها، عملگرها و جداکننده‌ها) را به فرمت ابزار انتخابی تبدیل کنید. حداقل سه برنامه‌ی ورودی برای ارزیابی تحلیل‌گر لغوی آماده کنید (شامل حداقل یک برنامه با خطای لغوی).

فاز سوم: تحلیل نحوی و تحلیل معنایی پایه

در این فاز گرامر MiniLang را به فرمت مورد قبول ابزار Yacc/Bison (یا معادل آن در زبان انتخابی) تبدیل کرده و پارسر را پیاده‌سازی کنید. سپس اعمال (actions) معنایی پایه را به گرامر اضافه کنید. خروجی این فاز باید برای هر برنامه‌ی ورودی، یکی از موارد زیر را اعلام کند: برنامه از نظر نحوی و معنایی صحیح است، یا پیغام خطای نحوی/معنایی مناسب (با ذکر محل خطا) نمایش داده شود.

بررسی‌های معنایی لازم:

- جدول نمادها (Symbol Table): یک جدول سراسری برای نگهداری متغیرهای تعریف‌شده و نوع آن‌ها کافی است (با حذف توابع، نیازی به جدول‌های جداگانه برای هر Scope نیست).
- تعریف قبل از استفاده: هر متغیر باید پیش از استفاده تعریف شده باشد و تعریف تکراری یک متغیر خطا محسوب می‌شود.
- تطابق نوع در انتساب: نوع دو طرف انتساب باید سازگار باشد؛ تبدیل خودکار bool → int یا string → int مجاز نیست.
- عبارات بولی: عملگرهای منطقی (&& , || , !) فقط روی مقادیر bool معتبرند و شرط if/while باید از نوع bool باشد.
- عملگرهای محاسباتی فقط روی مقادیر int تعریف شده‌اند.

حداقل ۳ برنامه‌ی تست (شامل حداقل یک برنامه با خطای نحوی و یک برنامه با خطای معنایی) به همراه خروجی آن‌ها باید تحویل داده شود.

فاز چهارم: تولید کد میانی

در این فاز رویه‌های تولید کد به گرامر اضافه می‌شوند تا برای هر برنامه‌ی ورودی MiniLang، کد سه‌آدرسه‌ی معادل (مطابق آنچه در کلاس درس استفاده شده) تولید شود. الزم است کد میانی برای موارد زیر به‌درستی تولید شود:

- ارزیابی عبارات محاسباتی، رابطه‌ای، تساوی و منطقی با رعایت اولویت عملگرها
- دستور انتساب
- دستور شرطی if / else با برچسب‌ها (labels) و پرش‌های مناسب
- حلقه‌ی while با برچسب‌های شروع، شرط و پایان حلقه
- دستورات print و input

برنامه باید خروجی هر مرحله (توکن‌ها، نتیجه‌ی تحلیل نحوی/معنایی و کد سه‌آدرسه) را برای حداقل ۳ برنامه‌ی ورودی نمونه نمایش دهد؛ نمایش در فایل خروجی یا خطفرمان کافی است و طراحی پنجره‌ی گرافیکی جداگانه الزامی نیست.

امتیازات اختیاری (Bonus)

گروه‌هایی که زمان و انگیزه‌ی کافی دارند می‌توانند برای کسب امتیاز اضافه، یک یا چند مورد از موارد زیر را به MiniLang اضافه کنند (هر مورد حداکثر ۰.۵ نمره‌ی اضافه، با تأیید استاد درس):

- افزودن نوع داده float و تبدیل ضمنی int → float

- افزودن آرایه‌های یک‌بعدی به‌همراه بررسی محدوده‌ی اندیس
- افزودن تعریف و فراخوانی تابع (func) با بررسی پارامترها و مقدار بازگشتی
- افزودن دستور for و عملگرهای رابطه‌ای <= و >=
- پشتیبانی از escape sequence در رشته‌ها

با آرزوی موفقیت