دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

اصول طراحی کامپایلر (دکتر ممتازی)

نیم‌سال اول سال تحصیلی ۱۴۰۱-۱۴۰۲

تمرین سری اول

دانشکده مهندسی کامپیوتر

قبل از حل سوالات به نکات زیر توجه فرمایید:

- هدف از انجام تمرین‌ها، یادگیری عمیق‌تر مطالب درسی است. در نتیجه هرگونه تقلب موجب کسر نمره خواهد شد.
- مهلت تحویل تا پایان روز ۲۸ آذر و نحوه تحویل از طریق سامانه کورسز است. (همه پاسخ‌ها را به صورت یک فایل فشرده ارسال کنید و نام فایل را شماره دانشجویی قرار دهید.)
- در صورت وجود هر گونه سوال می‌توانید از طریق ایمیل `CompilerFallAut2022@gmail.com` با تدریسیاران در ارتباط باشید.

---

۱. عبارت‌های منظم امکان تشخیص توضیحاتی [1] که برنامه‌نویس در برنامه خود نوشته است را ندارند؛ چرا که ممکن است توضیحات تو در تو باشند. زبانی را تصور کنید که به دو طریق `/* comment */` و `(* comment *)` می‌توان توضیحات درج کرد.

```
/*
    comment in depth 1
    (*
        comment in depth 2
        (*
            comment in depth 3
        *)
    *)
    another comment in depth 1
    /*
        another comment in depth 2
    */
*/
```

به کمک هر Lexical Analyzer Generator بر پایه Lex در یکی از زبان‌های Python، Java و C مانند کتاب‌خانه‌های PLY، SLY، JLex و Flex تحلیلگر لغوی بنویسید که کامنت‌ها را تشخیص داده و آنها را نادیده بگیرد و سایر ارقام و حروف انگلیسی را به

[1] comments

عنوان توکن [۲] خروجی دهد. در واقع برای سادگی فرض کنید زبان تنها شامل تعریف منظم $token \longrightarrow$ [0-9a-zA-Z] است. همچنین برنامه‌تان باید whitespace ها را نیز نادیده بگیرد. دقت کنید باز و بسته شدن کامنت‌ها باید با یکدیگر تطابق داشته باشد. به عنوان مثال `(* comment *)` `/* comment */` کامنت معتبری نیست ولی `/* (* comment *) */` یا `(* /* comment */ *)` معتبر هستند. برای نمونه برنامه زیر شامل ۸ توکن است.

```
1 hi1 2 /* good */
2 bye 7 (* bad /* inner */ *)
```

اگر از Flex یا JLex استفاده می‌کنید فایل با پسوند `.l` که به عنوان ورودی به Lexical Analyzer Generator داده‌اید را بارگذاری کنید. اگر از PLY یا SLY فایل ماژول یا کلاس Lexer که نوشتید را بارگذاری کنید.

۲. برای زبان‌های زیر اتوماتای متناهی قطعی [۳] ارائه دهید. برای هر استیت توضیحات ارائه دهید که مشخص شود کاربرد آن چیست. (منظور از $n_w(x)$ تعداد وقوع زیر رشته w در رشته x است. به عنوان مثال $n_{aa}(aaab) = ۲$ و $n_{ba}(abab) = ۱$.) ($\Sigma = \{a, b\}$)

(a) $L_1 = \{x \in \Sigma^* \mid n_{ab}(x) = n_{ba}(x)\}$.

(b) $L_2 = \{x \in \Sigma^* \mid n_a(x) = n_b(x)$ and for every prefix of x, $0 \leq n_a(x) - n_b(x) \leq 2\}$.

(c) $L_3 = \{x \in \Sigma^* \mid n_a(x)$ is even and $n_b(x)$ is odd$\}$.

(d) $L_4 = \{w \in (\Sigma \cup \{c\})^* \mid$ every a in w is followed by at least one b and at least one $c\}$. ($abaacb \in L_4$, $abacc \notin L_4$)

۳. برای هر یک از زبان‌های زیر، اگر زبان منظم است عبارت منظم آن را بیابید و سپس به صورت الگوریتمی، [۴] اتوماتای متناهی غیر قطعی [۵] متناظر با عبارت منظم را ساخته و با روش مطرح شده در کلاس [۶] آن را به اتوماتای متناهی قطعی تبدیل کنید. اگر زبان منظم نیست دلیل ارائه کنید. نیازی به اثبات نیست. ($\Sigma = \{a, b\}$)

(آ) زبان همه رشته‌هایی که دارای بیش از یک وقوع aa نباشند. (دقت کنید aaa دارای دو وقوع aa است.)

(ب) زبان همه رشته‌هایی که شامل زیر رشته bba نباشند.

(ج) زبان رشته‌هایی که در هر پیشوند آنها تعداد aها از تعداد bها کمتر نباشد.

(د) زبان همه رشته‌هایی که به ab ختم نمی‌شوند.

(ه) زبان همه رشته‌هایی که abb زیر دنباله آنها نباشد. (گوییم رشته w زیر دنباله x است اگر با حذف تعدادی از حروف x به w برسیم.)

[2] token
[3] Deterministic Finite Automata (DFA)
[4] Kleene's Algorithm
[5] Nondeterministic Finite Automata (NFA)
[6] subset construction

۴. گرامر زیر را در نظر بگیرید. (اولویت عملگرها از بیشترین به کمترین به ترتیب not سپس and و در نهایت or است. همچنین عملگرها right-associative هستند.)

$$E \longrightarrow E \textbf{ and } E$$

$$E \longrightarrow E \textbf{ or } E$$

$$E \longrightarrow (E)$$

$$E \longrightarrow \textbf{not } E$$

$$E \longrightarrow \textbf{id}$$

$$E \longrightarrow \textbf{true}$$

$$E \longrightarrow \textbf{false}$$

(آ) نشان دهید گرامر مبهم است. (درخت اشتقاق [۷] را رسم کرده و نوع اشتقاق [۸] را نیز مشخص کنید.)

(ب) گرامر را با توجه به اولویت‌ها و associativity رفع ابهام کنید. توضیح دهید که چرا گرامر حاصل ابهام ندارد.

۵. گرامر زیر را در نظر بگیرید. (مجموعه پایانه‌ها [۹] برابر است با $\{(,), +, *, \textbf{int}\}$ و A متغیر شروع گرامر است.)

$$A \longrightarrow \epsilon \mid A + B \mid C + C$$

$$B \longrightarrow C \mid C * B$$

$$C \longrightarrow (A) \mid \textbf{int}$$

(آ) گرامر دارای ساختار بازگشت چپ [۱۰] از چه نوعی می‌باشد؟ (مستقیم [۱۱] یا غیر مستقیم [۱۲]) بازگشت چپ را از گرامر حذف کنید.

(ب) در صورت نیاز تبدیل left factoring را روی گرامر بخش (آ) انجام دهید.

(ج) برای گرامر بخش (ب)، مجموعه‌های FIRST و FOLLOW را برای ناپایانه‌ها [۱۳] محاسبه کنید.

(د) برای گرامر بخش (ج)، جدول LL(1) را تشکیل دهید.

(ه) عبارت $(+\textbf{int} * \textbf{int}) + \textbf{int}$ را به کمک جدول LL(1) تجزیه کنید و مراحل تجزیه (قواعد استفاده شده و محتوای پشته) را مشخص کنید.

[7] derivation tree
[8] leftmost or rightmost
[9] terminals
[10] left recursion
[11] direct (immediate) left recursion
[12] indirect left recursion
[13] non-terminals

۶. (**امتیازی**) در یک عبارت منظم تعمیم یافته، فقط می‌توان از عملگرهای الصاق زبانی، [۱۴] مکمل [۱۵] و اشتراک [۱۶] مجموعه‌ای استفاده کرد. (در واقع از عملگر ستاره کلین [۱۷] نمی‌توان استفاده کرد ولی می‌توان از عملگر مکمل نسبت به مجموعه مرجع استفاده کرد.) به عنوان مثال $\varnothing^c\{ba\}\varnothing^c$ یک عبارت منظم تعمیم یافته است که زبان کلماتی را توصیف می‌کند که شامل زیر رشته ba هستند. (در واقع اگر مجموعه مرجع را $\{a,b\}^*$ فرض کنیم، $\varnothing^c$ مکمل مجموعه تهی نسبت به مجموع مرجع خواهد بود و یعنی $\varnothing^c = \{a,b\}^*$.) برای زبان $\{bab\}^*$ عبارت منظم تعمیم یافته ارائه دهید. (مجموعه مرجع را $\{a,b\}^*$ در نظر بگیرید.)

[14] concatenation
[15] complement
[16] intersection
[17] Kleene star