تکلیف سوم کامپایلر
تجزیه‌گرهای پایین به بالا

دانشگاه صنعتی اصفهان

پاییز ۹۸

سوالات تئوری:

۱. ابتدا برای رشته زیر یک اشتقاق مناسب بنویسید و سپس به وسیله ی آن رشته ی داده شده را به روش پایین به بالا تجزیه کنید؟ در هر مرحله بنویسید که از چندمین تبدیل از اشتقاق استفاده کردید، آیا ترتیب استفاده از تبدیل ها نظم خاصی دارد؟(دقت کنید که مجاز به کشید دی اف ای و یا هیچ روش دیگری نمی باشید.)

R→R | R // '|' is a terminal

R→RR

R→R*

$R \rightarrow R^+$

R→(R)

R→a

R→b

$ab(ba(a^+)^*)|b^+$

۲. گرامر زیر را در نظر بگیرید:

S → S S + | S S * | a

برای هر یک از رشته های زیر دستگیره را مشخص کنید.

1. SSS + a * +.
2. SS + a * a +.
3. aaa * a + +.

۳. قسمتی از یک گرامر که برای کامپایل یک زبان برنامه نویسی خاص استفاده می‌شود به شکل زیر است:

1. structure → id {body}
2. body → variable_list
3. instanceClass → id id
4. callVoidFuncVoid →id
5. function → id{fbody}
6. fbody → variable_list exp_list

7. variable_list → variableDef variable_list
8. variableDef → type id = integer
9. exp_list → exp exp_list
10. exp → exp operator exp | type id = exp | epsilon

توضیح قانون‌های گرامر:

قانون(rule) اول برای تولید structure تولید می‌شود.

قانون دوم برای تولید بدنه‌ی structure استفاده می‌شود.

قانون سوم برای نمونه گرفتن از structure استفاده می‌شود.

قانون چهارم برای صدا زدن توابع بدون ورودی و خروجی استفاده می‌شود.

قانون‌های بعدی برای تولید توابعی به کار می روند که نه ورودی و نه خروجی دارند.

به وسیله ی flex برنامه‌ای طراحی شده است که توکن‌های موجود را دقیقا به شکل موجود در گرامر به سمت bison پاس می‌دهد (برنامه‌ی flex ممکن است دوبار ورودی را بررسی کند).
با توجه به گرامر و توضیحات داده شده، آیا می‌توان این گرامر را به وسیله ی یک تجزیه‌گر پایین به بالا تجزیه کرد؟
اگر جواب مثبت است کلاس زیر را از پایین به بالا تجزیه کنید(دقت کنید که هیچ عضوی از structure را تغییر ندهید زیرا منجر به عوض شدن زبان می‌شود همچنین هنگام تجزیه کامنت ها را نادیده بگیرید).

```
Compiler{
        Int lex = 1          //initial value
        Int bison = 2        //initial value
        Int parser = 3       //initial value
}
```

اگر جواب منفی است، دلیل بیاورید و بگویید چگونه می‌توان آن را رفع کرد؟(راه حل شما می تواند در سمت lexical analyzer و یا در سمت syntax analyzer باشد)

۴. در تجزیه گر پایین به بالا ، رفتار تجزیه‌گر به چه مواردی وابسته است؟

رشته ی زیر را با استفاده از رسم automaton LR0 و به دست آوردن جدول پارس، تجزیه کنید؟

S→Ba|EBc|Ac|EAa
E→dE|d
A→b
B→b

رشته ی مورد نظر:

ddddba

۵. گاهی با وجود تداخل در یک جدول تجزیه می توان با در نظر گرفتن روابط تقدم یا انجمنی میان عملگرها برای گرامر تجزیه کننده پایین به بالا ایجاد کرد. برای گرامر زیر یک جدول تجزیه SLR(1) ایجاد کنید. در صورت تداخل سعی کنید از روابط تقدم و در نظر گرفتن انجمنی عملگرها، تداخل را برطرف کنید.

R→R | R
R→RR
R→R*
R→R+
R→(R)
R→a
R→b

۶. گرامر زیر، دسته ای از گرامرها را تعریف می‌کند:

$S \rightarrow A_i\, b_i$ for $1 \le i \le n$

$A_i \rightarrow a_j\, A_i \mid a_j$ for $1 \le i, j \le n$ and $i \ne j$

الف) نشان دهید که گرامر دارای $2^n + n^2 + n$ مجموعه آیتم LR(0) است.

ب) آیا گرامر SLR(1) است؟

۷. نشان دهید که گرامر زیر LR(1) است اما LALR(1) نیست.

S → A a | b A c | B c | b B a
A → d

B → d

۸. گرامر زیر را در نظر بگیرید. فرض کنید e و s پایانه هستند.

stmt → if e then stmt
| if e then stmt else stmt
| while e do stmt
| begin list end
| s
list → list; stmt
| stmt

الف) یک جدول پایین به بالا برای این گرامر رسم کنید به طوری که conflict ها در مورد else هم در آن برطرف شده باشد.

ب) با استفاده از روش‌هایی که در کلاس خواندید فضای خالی پارسر را پر کنید تا بتوانیم خطاها را کنترل کنیم.

ج) با استفاده از روش کنترل خطای panic عملیات پارسر را بر روی عملیات زیر انجام دهید.

(i) if e then s; if e then s end

(ii) while e do end s; if e then s; end

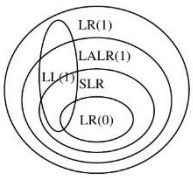
سوال اختیاری:

۱. شکل زیر ارتباط گرامر های مختلف را نشان می دهد. از شکل زیر چه نتایجی به دست می آورید؟ با در نظر گرفتن این شکل به پرسش های زیر پاسخ دهید:

الف) نشان دهید هر گرامر LL(1) یک گرامر R(1).

ب) نشان دهید هر گرامر LL(k) یک گرامر LR(k) است.

پ) نشان دهید یک گرامر LR(1) نمی تواند مبهم باشد.



سوالات عملی:

با استفاده از bison یک ماشین حساب بسازید که عملیات‌های زیر را بتواند انجام دهد:

۱. اگر اسم یک متغیر را دریافت کرد، مقدار آن را نمایش دهد.
۲. اگر یک عبارت ریاضی دریافت کرد آن را حساب کند و نتیجه را چاپ کند.
۳. اگر یک مقداردهی به یک متغیر دریافت کرد، مقدار آن را حساب کند و در متغیر ذخیره کند به گونه‌ای که از مقدار متغیر بتوان در خطوط بعد استفاده کرد(دقت کنید در این حالت نیازی به چاپ نتیجه ی خروجی نیست)
۴. از عملیات های ضرب، تقسیم، جمع و تفریق پشتیبانی کند.
۵. از توابع exp, log, sqrt پشتیبانی کند.

نکته۱: اعداد می توانند اعشاری و یا صحیح باشند.

نکته۲: متغیرها فقط می توانند حروف کوچک انگلیسی باشند.

گرامر نمونه که البته در صورت نیاز باید آن را تغییر دهید و یا این که از یک گرامر دیگر استفاده کنید:

line → line '\n' line | ID = line | line + line | line * line | line / line | line – line | (line) | NUMBER | DOUBLE | ID | function (line)

function → sqrt | exp | log

مثال:

ورودی:

```
a = 3 * 4 + (4 / 2 + 3)
4 * 5 – 2       //produce first output
b = 1 / 10
c = a * b
c               //produce second output
d = log(exp(4))
```

خروجی:

18

1.7