# سوال ۱

آ) آنالیز معنایی

ب) پیش پردازش

ج) آنالیز خطی[[1]](#footnote-1)

د) اشتقاق چپ به راست

ه) آنالیز خطی

و) جدول نمادها[[2]](#footnote-2)

ز) گاهی برای پردازش یک نشانه و انجام عمل کاهش نیاز داریم که این کار را با توجه به نشانه‌های بعدی انجام دهیم.

ح) این حرس دقیقا همان عمل کاهش در یک تحلیلگر لغوی پایین به بالا می‌باشد.

# سوال ۲

آ) از استفرا روی تعداد گره‌های درخت اشتقاق استفاده می‌کنیم، فرض می‌کنیم برای هر رشته از این زبان که درخت اشتقاق آن n گره دارد آن رشته بر ۳ بخش پذیر باشد. اگر فرض کنیم تعداد این اشتقاق‌ها n + 1 باشد در این صورت اولین اشتقاق در درخت را در نظر می‌گیریم، این اشتقاق به یکی صورت‌های زیر می‌باشد:

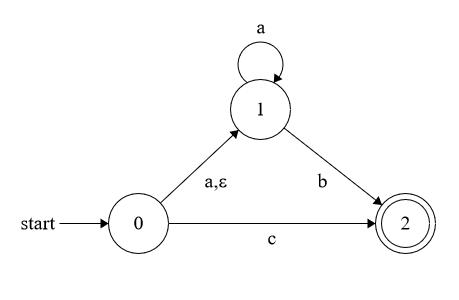
که هر دوی این اشتقاق‌ها اگر num بر ۳ بخش پذیر باشد، بر ۳ بخش پذیر می‌ماند.

ب) رشته‌ی زیر بر ۳ بخش پذیر می‌باشد و در این زبان طول نمی‌شود.

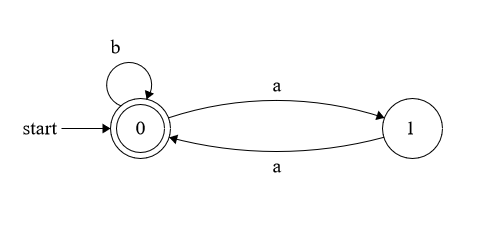
# سوال ۳

آ)

ب)



ج)



د) خیر – این زبان دارای شکلی به صورت پس باید تعداد a های اول و آخر به هم وابسته هستند پس نمی‌تواند عبارت منظم باشد.

ه)

# سوال ۴

آ) می‌دانیم با استفاده از قواعد منظم می‌توانیم یک NFA برای زبان تولید نماییم، در صورتی که این NFA را در جهت عکس پیمایش کنیم می‌توانیم رشته مورد نظر را به صورت عکس بپذیریم، تنها مشکل در صورتی رخ می‌دهد که NFA چند state برای اتمام داشته باشد، که به این ترتیب NFA جدید چند state برای شروع خواهد داشت، برای حل این مشکل یک state شروع اضافه می‌کنیم که با به stateهای شروع اصلی می‌رود.

ب) اگر رشته به گونه‌ای باشد که در پیمایش چپ و راست به رشته‌هایی برخورد کنیم که هر یک به نشانه‌ی مختلفی کاهش یابند چیزی که می‌خواهیم رخ می‌دهد:

از چپ به راست:

از راست به چپ:

# سوال ۵

آ) از آنجایی که این گرامر تکرار چپ در قواعد خود دارد در صورتی که به صورت LL(1) نوشته شود یک بازگشت بی‌نهایت رخ می‌دهد زیرا ورودی خوانده نمی‌شود و به صورت ممتد تابع A() فراخوانی می‌گردد.

ب)

ج)

void S() {

int isave = input\_pointer;

if (input\_symbol == 'a') {

advance();

return TRUE;

}

input\_pointer = isave;

if (input\_symbol == '(') {

advance();

if (A()) {

if (input\_symbol == ')') {

advance();

return TRUE;

} else {

input\_pointer = isave;

return FALSE;

}

} else {

if (input\_symbol == ')') {

advance();

return TRUE;

} else {

input\_pointer = isave;

return FALSE;

}

}

}

}

# سوال ۶

آ)

ب)

ج)

# سوال ۷

آ) مقادیر اعداد زیر را به ترتیب چاپ می‌کند

MAXINT-5

MAXINT-4

MAXINT-3

MAXINT-2

MAXINT-1

ب)

def new\_temp():

pass

def new\_label():

pass

def gen():

pass

def for\_loop(stmt\_1, v, initial, final, stmt\_2):

'''

stmt\_1 -> for v := initial to final do stmt\_2

'''

# Temporary storage for initial

t1 = {}

t1.place = new\_temp()

# Temporary storage for final

t2 = {}

t2.place = new\_temp()

stmt\_1.begin = new\_label()

stmt\_1.after = new\_label()

stmt\_1.while\_begin = new\_label()

stmt\_1.in\_while\_begin = new\_label()

stmt\_1.code = gen(t1.place + ":=" + initial.place) +\

gen(t2.place + ":=" + final.place) +\

gen("if" + t1.place + "<=" + t2.place + "goto" + stmt\_1.begin) +\

gen("goto" + stmt\_1.after) +\

gen(stmt\_1.begin + ":") + gen(v.place + ":=" + t1.place) +\

stmt\_2.code +\

gen(stmt\_1.while\_begin + ":") +\

gen("if" + v.place + "<>" + t2.place + "goto" + stmt\_1.in\_while\_begin) +\

gen("goto" + stmt\_1.after) + gen(stmt\_1.in\_while\_begin + ":") +\

gen(v.place + ":=" + "succ(" + v.place + ")") +\

stmt\_2.code +\

gen("goto" + stmt\_1.while\_begin) + gen(stmt\_1.after + ":")

# سوال ۸

نحوه‌ی تولید کد میانی با توجه به این پارسر هنوز تدریس نشده است. گرامر با حذف تکرار چپ در ادامه آمده است:

1. Lexical Analysis [↑](#footnote-ref-1)
2. Symbol Table [↑](#footnote-ref-2)