به نام خداوند بخشنده مهربان

فاز ۲ پروژه - درس اصول و طراحی کامپایلرها دکتر رزازی ترم پاییز ۱۴۰۰-۱۳۹۹ - دانشکده کامپیوتر، دانشگاه صنعتی امیرکبیر	

لطفا قبل از شروع به حل کردن تمرین به نکات زیر توجه فرمایید:

۱. هدف از انجام تمرین‌ها، یادگیری عمیق‌تر مطالب درسی است. در نتیجه هرگونه کپی‌برداری موجب کسر نمره خواهد شد.

۲. تمام فایل‌های خواسته شده را در یک فایل فشرده قرار دهید. نام فایل نهایی را شماره دانشجویی خود قرار دهید. (برای مثال parser_Family_9631747.zip)

۳. در کنار این فایل ۲ ویدئو برای راهنمایی شما وجود دارد که توصیه می کنیم آن ها را مشاهده بکنید. محتوای ارائه شده در فیلم ها در آدرس زیر قابل دسترسی است:

https://drive.google.com/drive/folders/1fzZaRvMtPemIPsTW8BiuZ7qNUBNQSZTL

۴. لطفا ۳ فایل متنی بارگزاری شده را به عنوان ورودی در نظر بگیرید و خروجی آن را در فایل ها جداگانه به همراه کد های برنامه در فایل فشرده قرار بدهید.

۵. در صورت وجود هرگونه سوال می توانید از طریق ایمیل با تدریسیار در ارتباط باشید.

moh.robati@aut.ac.ir

sheykh@aut.ac.ir

rouzbehghasemi1998@gmail.com

sepehr.asgarian@gmail.com

amirali.sajjadi98@gmail.com

parsafarinnia@gmail.com

شما در مرحله قبل یک تحلیل گر لغوی[1] طراحی کردید که لغات[2] را در برنامه تشخیص داده و نشانه[3] مربوط به آن را برگرداند.حال ما از شما می خواهیم یک تجزیه گر[4] طراحی کنید که از نشانه های برگردانده شده استفاده کند ،به طوریکه با قوانین گرامر داده شده مطابق باشد.

نکته:گرامر داده شده ابهام[5] دارد و شما باید با استفاده از اولویت[6] ها و تغییر گرامر ، بدون اینکه زبان پذیرنده زبان عوض شود، رفع ابهام کنید .

Grammar:

program → declist main ( ) block

declist → dec | declist dec | ϵ

dec → vardec | funcdec

type → int | float | bool

iddec → id | id [exp] | id=exp

idlist → iddec | idlist , iddec

vardec → idlist :type ;

funcdec → fun (paramdecs): type block | fun id (paramdecs) block

paramdecs → paramdecslist | ϵ

paramdecslist → paramdec | paramdecslist , paramdec

paramdec → id : type | id [] : type

[1] Lexical analyzer
[2] Lexemes
[3] Token
[4] Parser
[5] Ambiguity
[6] Precedence

block → { stmtlist }

stmtlist → stmt | stmlist stmt | ϵ

lvalue → id | id [exp]

case → where const: stmtlist

cases → case | cases case | ϵ

stmt → return exp ; | exp ;| block | vardec |
while (exp) stmt | on (exp) {cases }; |
for(exp ; exp ; exp) stmt | for(id in id) stmt |
if (exp) stmt elseiflist | if (exp) stmt elseiflist else stmt |
print (id) ;

elseiflist → elseif (exp) stmt | elseiflist elseif (exp) stmt | ϵ

relopexp → exp relop exp | relopexp relop exp

exp → lvalue=exp | exp operator exp | relopexp |
const | lvalue | id(explist) | (exp) | id() | - exp | not exp

operator → and | or | + | - | * | / | %

const → intnumber | floatnumber | True | False

relop → > | < | != | == | <= | >=

explist → exp | explist,exp