CS305 – Programming Languages
Fall 2020-2021

Homework 1

Due date: October 18, 2020 @ 23:55

Implementing a Lexical Analyzer (Scanner) for MailScript

1 Introduction

In this homework you will implement a scanner for ***MailScript*** language using flex. Your scanner will be used to produce the tokens in a MailScript program. This is a scripting language that will be used for e-mail automation.

MailScript is loosely based on AppleScript, which is a scripting language that facilitates automated control over scriptable Mac applications. MailScript uses a block structure. Each block starts with the keywords "**Mail from**" followed by the e-mail of the user and a colon symbol. Each block ends with the keywords "**end Mail**". One can use the keyword "**send**" in order to send e-mails to certain users. Also they can create a sub-block called "**schedule**" and specify a certain date for the e-mails to be sent. One can "**set**" variables in and out of block. An example MailScript file is given below:

```
set Message ("Welcome to CS305.")
set Name ("Deniz")

Mail from derya@mail.com:

    send ["Hello!"] to [("Ayse", ayse@mail.com),
    (mehmet@mail.com), ("Mehmet", mehmet@mail.com)]

    schedule @ [03/10/2021, 16:00]:
```

```
        send [Message] to [("Beril", beril@mail.com.tr)]
        send ["Thank you
        very much."] to [(Name, deniz@mail.com.tr)]
    end schedule

end Mail


Mail from derya@sabanciuniv.edu:

    schedule @ [02/10/2021, 16:00]:
        send ["Good morning!"] to [(ali@mail.com),
        ("Ferhat Kaya", ferhat@mail.com),
        ("Ali", ali@mail.com)]
    end schedule

end Mail

Mail from derya@mail.com:

    schedule @ [03/10/2021, 04:00]:
        send ["How are you?"] to [("Omer", omer@mail.com)]
    end schedule

end Mail
```

In a MailScript program, there may be keywords (e.g. send, set, ...), values (e.g. e-mails, dates, ...), identifiers (programmer defined names for variables, etc.) and punctuation symbols (like , : etc.). Your scanner will catch these language constructs (introduced in Sections 2, 3, 4) and print out the token names together with their positions (explained in Section 5) in the input file. Please see Section 6 for an example of the required output. The following sections provide extensive information on the homework. Please read the entire document carefully before starting your work on the homework.

2 Keywords

Below is the list of keywords to be implemented, together with the corresponding token names.

Lexeme	**Token**	**Lexeme**	**Token**
Mail	tMAIL	end Mail	tENDMAIL
schedule	tSCHEDULE	end schedule	tENDSCH
send	tSEND	to	tTO
from	tFROM	set	tSET

MailScript is case–sensitive. Only the lexemes given above are used for the keywords.

3 Operators & Punctuation Symbols

Below is the list of operators and punctuation symbols to be implemented, together with the corresponding token names.

Lexeme	**Token**	**Lexeme**	**Token**
,	tCOMMA	:	tCOLON
(	tLPR	)	tRPR
[	tLBR	]	tRBR
@	tAT		

4 Identifiers & Values

You need to implement identifiers and values (strings, e-mails, dates and time). Here are some rules you need to pay attention to:

- An identifier consists of any combination of letters, digits and underscore character. However, it cannot start with a digit.
- The token name for an identifier is tIDENT.
- Anything inside a pair of quotation marks is considered as a string. However a string cannot contain a quotation mark itself. The empty string (an opening quote immediately followed by a closing quote) is also a string. The token name for a string is tSTRING.
- An e-mail address is made up of a local-part, an @ sign and a domain. It can be represented as local-part@domain.
- The local-part can only contain uppercase and lowercase Latin letters, digits, hyphens (-), underscores (_) and dots (provided that they cannot be the first or the last characters of the local-part and cannot appear consecutively). For example, the following lexemes would NOT be recognized as an e-mail address: .example@mail.com, example.@mail.com, ex..ample@mail.com.
- The domain name should consist of 2 or 3 dot-separated labels: xxx.xxx or xxx.xxx.xxx. Each label can contain uppercase and lowercase Latin letters, digits and hyphens (-), provided that it is not the first or last character. For example, the following lexemes would NOT be recognized as an e-mail address: example@-mail.com, example@mail-.com.
- The token name for an e-mail address is tADDRESS. Some examples are given below:

```
example@mail.com
john.smith@new-mail.com.tr
User--123@a-really-long-domain.name
12345@12345.12345
```

- A date will be given in the following format: DD/MM/YYYY (e.g. 11/10/2021). All day, month and year slots can take any digit. Hence the following example would also be considered a date variable: 45/98/0290. The token name for a date variable is tDATE.

- A time variable will be given in the following format: HH:MM (e.g. 16:30). All hour and minute slots can take any digit. Hence the following example would also be considered a time variable: 25:90. The token name for a time variable is tTIME.
- Any character which is not a whitespace character and which cannot be detected as the part of lexeme of a token should be printed out together with an error message.

5 Positions

You must keep track of the position information for the tokens. For each token that will be reported, the line number at which the lexeme of the token appears is considered to be the position of the token. Please see Section 6 to see how the position information is reported together with the token names.

6 Input, Output, and Example Execution

Assume that your executable scanner (which is generated by passing your flex program through flex and by passing the generated lex.yy.c through the C compiler gcc) is named as MSScanner. Then we will test your scanner on a number of input files using the following command line:

```
MSScanner < test17.ms
```

As a response, your scanner should print out a separate line for each token it catches in the input file (test17.ms given in Figure 1). The output format for a token is given below for each token separately:

Token	**Output**
identifier, date, time, string, e-mail addr.	⟨*row*⟩⟨*space*⟩⟨*token_name*⟩⟨*space*⟩(⟨*lexeme*⟩)
for all other tokens	⟨*row*⟩⟨*space*⟩⟨*token_name*⟩
for illegal characters	⟨*row*⟩⟨*space*⟩ ILLEGAL CHARACTER ⟨*space*⟩(⟨*lexeme*⟩)

Here, ⟨*row*⟩ gives the location (line number) of the first character of the lexeme of the token and ⟨*token_name*⟩ is the token name for the current item. ⟨*lexeme*⟩ will display the lexeme of the tokens of type identifier (tIDENT), date (tDATE), time (tTIME), string (tSTRING) and e-mail address (tADDRESS). And ⟨*space*⟩ corresponds to a single space. **Also, please note that you are supposed to print the lexeme of the tSTRING token without the quotation marks as given in the examples below.** For example let us assume that test17.ms has the following content:

```
Mail from beril@mail.com:

    set News ("Hi, I will not be joining today's meeting.")
    schedule @ [19/12/2021, 08:30]:
        send [News] to [("Tugce", tugce@company-mail.com)]
    end schedule

end Mail
```

Figure 1: An example MailScript program: test17.ms

Then the output of your scanner must be:

```
1 tMAIL
1 tFROM
1 tADDRESS (beril@mail.com)
1 tCOLON
3 tSET
3 tIDENT (News)
3 tLPR
3 tSTRING (Hi, I will not be joining today's meeting.)
3 tRPR
4 tSCHEDULE
4 tAT
4 tLBR
```

```
4 tDATE (19/12/2021)
4 tCOMMA
4 tTIME (08:30)
4 tRBR
4 tCOLON
5 tSEND
5 tLBR
5 tIDENT (News)
5 tRBR
5 tTO
5 tLBR
5 tLPR
5 tSTRING (Tugce)
5 tCOMMA
5 tADDRESS (tugce@company-mail.com)
5 tRPR
5 tRBR
6 tENDSCH
8 tENDMAIL
```

Note that, the content of the test files need not be a complete or correct MailScript program. If the content of a test file is the following:

```
:: schedule ]Ali @ from Mail
example@mail.edu
send
99/99/9999
.example@mail.com
example@-mail.com
```

Figure 2: An example MailScript program: test18.ms

Then your scanner should not complain about anything and output the following information:

```
1 tCOLON
1 tCOLON
1 tSCHEDULE
1 tRBR
1 tIDENT (Ali)
1 tAT
1 tFROM
1 tMAIL
2 tADDRESS (example@mail.edu)
3 tSEND
4 tDATE (99/99/9999)
5 ILLEGAL CHARACTER (.)
5 tADDRESS (example@mail.com)
6 tIDENT (example)
6 tAT
6 ILLEGAL CHARACTER (-)
6 tIDENT (mail)
6 ILLEGAL CHARACTER (.)
6 tIDENT (com)
```

7 How to Submit

Submit only your flex file (**without zipping it**) on SUCourse. The name of your flex file must be:

username-hw1.flx

where username is your SuCourse username.

8 Notes

- **Important**: Name your file as you are told and **don't zip it.** **[-10 points otherwise]**
- Do not copy-paste MailScript program fragments from this document as your test cases. Copy/paste from PDF can create some unrecognizable characters. Instead, all MailScript codes fragments that appear in this document are provided as a text file for you to use.

- Make sure you print the token names exactly as it is supposed to be. You will lose points otherwise.
- No homework will be accepted if it is not submitted using SUCourse+.
- You may get help from our TA or from your friends. However, **you must write your flex file by yourself**.
- Start working on the homework immediately.
- If you develop your code or create your test files on your own computer (not on flow.sabanciuniv.edu), there can be incompatibilities once you transfer them to flow.sabanciuniv.edu. Since the grading will be done automatically on the flow.sabanciuniv.edu, we strongly encourage you to do your development on flow.sabanciuniv.edu, or at least test your code on flow.sabanciuniv.edu before submitting it. If you prefer not to test your implementation on flow.sabanciuniv.edu, this means you accept to take the risks of incompatibility. Even if you may have spent hours on the homework, you can easily get 0 due to such incompatibilities.
- LATE SUBMISSION POLICY:
  Late submission is allowed subject to the following conditions:
  - Your homework grade will be decided by multiplying what you get from the test cases by a "submission time factor (STF)".
  - If you submit on time (i.e. before the deadline), your STF is 1. So, you don't lose anything.
  - If you submit late, you will lose 0.01 of your STF for every 5 mins of delay.
  - We will not accept any homework later than 500 mins after the deadline.
  - SUCourse+'s timestamp will be used for STF computation.
  - If you submit multiple times, the last submission time will be used.