CS305 – Programming Languages

Spring 2024-2025

HOMEWORK 2

Implementing a Syntax Analyzer (Parser) for Meeting Scheduler (MS)

Due date: March 16th, 2025 @ 23:55

NOTE

Only SUCourse submissions are allowed. Submissions by e-mail are not allowed. Please see the note at the end of this document for the late submission policy.

1 Introduction

In this homework, you will write a context-free grammar and implement a parser for (MS) language, which you designed as a scanner in the previous homework. The language that will be generated by your grammar and other homework requirements are explained below.

2 Meeting Scheduler (MS) Language

The grammar you will design needs to generate the Meeting Scheduler (MS) language described below. Here is an example program written in this language to give you an idea of what an MS program looks like.

```
1     Meeting "Meeting with the employees"
2         meetingNumber = 1257
3         description = "Month-end Reports"
4         startDate = 24.03.2025
5         startTime = 14.40
6         endDate = 25.03.2025
7         endTime = 16.30
8         locations = FENSG032, FENSG035
9         isRecurring = yes
10        frequency = monthly
11        repetitionCount = 12
12        subMeetings
13            Meeting "Meeting with Engineers"
14                meetingNumber = 222
```

```
        description = "Month-end Reports of Engineers"
        startDate = 24.03.2025
        startTime = 14.40
        endDate = 24.03.2025
        endTime = 16.30
        locations = FENSG032, FENSG035
        isRecurring = yes
        frequency = monthly
        repetitionCount = 12
        subMeetings
            Meeting "Meeting with Alice"
                meetingNumber = 2568
                description = "Month-end Reports of Alice"
                startDate = 24.03.2025
                startTime = 14.40
                endDate = 24.03.2025
                endTime = 15.30
                locations = FENSG032
                isRecurring = yes
                frequency = monthly
                repetitionCount = 12
            endMeeting

            Meeting "Meeting with Bob"
                meetingNumber = 4
                description = "Month-end Reports of Bob"
                startDate = 24.03.2025
                startTime = 15.40
                endDate = 24.03.2025
                endTime = 16.30
                locations = FENSG032
                isRecurring = yes
                frequency = monthly
                repetitionCount = 12
            endMeeting
        endSubMeetings
    endMeeting

    Meeting "Meeting with Accountants"
        meetingNumber = 432
        description = "Month-end Reports of Accountants"
        startDate = 25.03.2025
        startTime = 14.40
        endDate = 25.03.2025
        endTime = 16.30
```

```
                locations = FENSG035
                isRecurring = yes
                frequency = monthly
                repetitionCount = 12
                subMeetings
                    Meeting "Meeting with George"
                        meetingNumber = 2568
                        description = "Month-end Reports of George"
                        startDate = 25.03.2025
                        startTime = 14.40
                        endDate = 25.03.2025
                        endTime = 15.30
                        locations = FENSG035
                        isRecurring = yes
                        frequency = monthly
                        repetitionCount = 12
                    endMeeting

                    Meeting "Meeting with Hagi"
                        meetingNumber = 4
                        description = "Month-end Reports of Hagi"
                        startDate = 25.03.2025
                        startTime = 15.40
                        endDate = 25.03.2025
                        endTime = 16.30
                        locations = FENSG035
                        isRecurring = yes
                        frequency = monthly
                        repetitionCount = 12
                    endMeeting
                endSubMeetings
            endMeeting
        endSubMeetings
    endMeeting
```

Here is the description of the syntactic rules of MS language:

1. An MS program consists of a non–empty list of meeting blocks. In other words, there must be at least one meeting block in each MS program.

2. Each meeting block starts with the keyword of **Meeting** followed by a string (which is the name of the meeting) and ends with the keyword **endMeeting**. Inside a meeting block, we have the following elements:

    - "meetingNumber" element (required):
      This element starts with the keyword **meetingNumber**. It is followed by an equality symbol and an integer.

- "description" element (required):
  This element starts with the keyword **description**. It is followed by an equality symbol. Finally a string must exist.
- "startDate" element (required):
  This element starts with the keyword **startDate**. It is followed by an equality symbol. Finally a date expression must be given.
- "startTime" element (required):
  This element starts with the keyword **startTime**. It is followed by an equality symbol. Finally a time expression must be given.
- "endDate" element (required):
  This element starts with the keyword **endDate**. It is followed by an equality symbol. Finally a date expression must be given.
- "endTime" element (required):
  This element starts with the keyword **endTime**. It is followed by an equality symbol. Finally a time expression must be given.
- "locations" element (required):
  This element starts with the keyword **locations**. It is followed by an equality symbol. Finally a non–empty list of comma–separated locations are given, where each location is syntactically an identifier.
- "isRecurring" element (required):
  This element starts with the keyword **isRecurring**. It is followed by an equality symbol. Finally, we must either have **yes** or **no** expression provided.
- "frequency" element (optional):
  This element starts with the keyword **frequency**. It is followed by an equality symbol. Finally, we must have either **daily**, **weekly**, **montly** or **yearly** specified.
- "repetitionCount" element (optional):
  This element starts with the keyword **repetitionCount**. It is followed by an equality symbol. Finally a non–negative integer must be given.
- "subMeetings" element (optional):
  This element starts with the keyword **subMeetings** and ends with the keyword **endSubMeetings**. Between these two keywords, we again have a non–empty list of meeting blocks, just like what we have for an entire MS program.

Some important notes about the grammar:

- The order of the elements given above is fixed, i.e. they must appear inside a meeting block in the order given above. In other words, inside a meeting block, we always have the meetingNumber element first, and then we have the description element, etc.
- The required elements must exist and they exist only once in each meeting block.

- The optional elements may or may not exist in a meeting block. If they appear, they appear in an order given above. That is, for example, we cannot have repetition count before frequency. In addition, there cannot be multiple occurences of these optional elements. In other words, we can have at most one frequency element, we can have at most one repetition count element, etc.

- The time and the date expressions may not correspond to an actual time and date. Hence we still allow 56.72.1010 as a date expression, and 34.93 as a time expression.

- The frequency and the repetetionCount elements are optional. It may seem unnecessary to have the frequency and repetitionCount elements if the meeting is not recurring. Conversely, it may seem that we need to have the frequency and the repetetionCount elements existing, if the meeting is recurring. However, for the purposes of this homework, where we will only check the grammatical correctness, such concerns are not valid. Therefore, for example, we will accept the following meeting blocks as grammatically correct as well:

```
Meeting "Strange Example 1"
   meetingNumber = 1258
   description = "Recurring meeting with no frequency/repetition"
   startDate = 24.03.2025
   startTime = 14.40
   endDate = 25.03.2025
   endTime = 16.30
   locations = FENSG032
   isRecurring = yes
endMeeting

Meeting "Strange Example 2"
   meetingNumber = 1259
   description = "Non--recurring meeting with frequency/repetition"
   startDate = 24.03.2025
   startTime = 14.40
   endDate = 25.03.2025
   endTime = 16.30
   locations = FENSG035
   isRecurring = no
   frequency = monthly
   repetitionCount = 12
endMeeting

Meeting "Strange Example 3"
   meetingNumber = 1260
   description = "Recurring meeting with only frequency"
   startDate = 24.03.2025
```

```
29          startTime = 14.40
30          endDate = 25.03.2025
31          endTime = 16.30
32          locations = FENSG035
33          isRecurring = yes
34          frequency = daily
35        endMeeting
36
37      Meeting "Strange Example 4"
38          meetingNumber = 1261
39          description = "Recurring meeting with only repetitionCount"
40          startDate = 24.03.2025
41          startTime = 14.40
42          endDate = 25.03.2025
43          endTime = 16.30
44          locations = FENSG035
45          isRecurring = yes
46          repetitionCount = 12
47        endMeeting
```

- The subMeetings element is also optional. However, if the subMeetings element exists, we will have again a non–empty list of meeting blocks in it. So, between subMeetings and endSubMeetings keywords, we again have an entire MS program. Note that, this implies, for a meeting block inside a subMeetings element, we can also have subMeetings (see the first example given in this document). Furthermore, there is no restriction on the number of levels of subMeetings. Hence we can have subMeetings of subMeetings of subMeetings of ..... (and we can go like this indefinitely).

3 Terminal Symbols

Although you can implement your own scanner, we provide a flex scanner for this homework. The provided flex scanner implements the following tokens.

`tSTARTMEETING`: The scanner returns this token when it sees a `Meeting` keyword.

`tENDMEETING`: The scanner returns this token when it sees an `endMeeting` keyword.

`tSTARTSUBMEETINGS`: The scanner returns this token when it sees a `subMeetings` keyword.

`tENDSUBMEETINGS`: The scanner returns this token when it sees an `endSubMeetings` keyword.

`tMEETINGNUMBER`: The scanner returns this token when it sees a `meetingNumber` keyword.

`tDESCRIPTION`: The scanner returns this token when it sees a `description` keyword.

`tSTARTDATE`: The scanner returns this token when it sees a `startDate` keyword.

`tSTARTTIME`: The scanner returns this token when it sees a `startTime` keyword.

`tENDDATE`: The scanner returns this token when it sees an `endDate` in the input.

`tENDTIME`: The scanner returns this token when it sees an `endTime` keyword.

`tLOCATIONS`: The scanner returns this token when it sees a `locations` keyword.

`tISRECURRING`: The scanner returns this token when it sees an `isRecurring` keyword.

`tFREQUENCY`: The scanner returns this token when it sees a `frequency` keyword.

`tREPETITIONCOUNT`: The scanner returns this token when it sees a `repetitionCount` keyword.

`tDAILY`: The scanner returns this token when it sees a `daily` keyword.

`tWEEKLY`: The scanner returns this token when it sees a `weekly` keyword.

`tMONTHLY`: The scanner returns this token when it sees a `monthly` keyword.

`tYEARLY`: The scanner returns this token when it sees a `yearly` keyword.

`tYES`: The scanner returns this token when it sees a `yes` keyword.

`tNO`: The scanner returns this token when it sees a `no` keyword.

`tASSIGN`: The scanner returns this token when it sees an `=` symbol.

`tCOMMA`: The scanner returns this token when it sees a `,` symbol.

`tIDENTIFIER`: The scanner returns this token when it sees an identifier.

`tSTRING`: The scanner returns this token when it sees a string.

`tINTEGER`: The scanner returns this token when it sees a non-negative integer.

`tDATE`: The scanner returns this token when it sees a date expression with the pattern DD.MM.YYYY (no check is performed if it is an actual valid date, hence even 45.52.1111 would return a `tDATE` token).

`tTIME`: The scanner returns this token when it sees a time expression with pattern HH.MM (no check is performed if it is an actual valid time, hence even 42.83 would return a `tTIME` token.

Besides these tokens, the scanner silently consumes white space characters. Any other character that is not recognized as part of a lexeme of a token is returned to the parser.

These tokens and their lexemes are explained in detail in the Homework 1 document.

4 Output

Your parser must print out `OK` and produce a new line if the input is grammatically correct. Otherwise, your parser must print out `ERROR` and produce a new line. In other words, the main part in your parser file must be as follows (and there should be no other part in your parser that produces an output):

```
int main ()
{
   if (yyparse())
   {
      // parse error
      printf("ERROR\n");
      return 1;
   }
   else
   {
      // successful parsing
      printf("OK\n");
      return 0;
   }
```

```
}
```

In short, if the file `example1.ms` includes a grammatically correct MS program then your output should be `OK`, and otherwise, your output should be `ERROR`.

5 How to Submit

You need to submit:

1. your Bison file
2. your flex file (**IMPORTANT**: Even if you use the flex file we provided, you still need to submit it after renaming it as indicated below.)

Please submit both of these files (**without zipping them**) on SUCourse. The name of your bison file must be **username-hw2.y**, and the name of your flex file must be **username-hw2.flx** where **username** is your SU-Net username.

We will compile your files by using the following commands:

```
flex username-hw2.flx
bison -d username-hw2.y
gcc -o username-hw2 lex.yy.c username-hw2.tab.c -lfl
```

So, ensure these three commands are enough to produce the executable parser. If we assume that there is a text file named example1.ms, we will try out your parser by using the following command line:

```
username-hw2 < example1.ms
```

6 Notes

- **Important**: Name your files as you are told and **don't zip them.** [-20 points otherwise]
- **Important**: Make sure you include the correct "`...tab.h`" file in your scanner, and make sure you can compile your parser using the commands given in Section 5. If we are not able to compile your code with those commands **your grade will be zero for this homework.**
- **Important**: Since this homework is evaluated automatically, ensure your output is exactly as it should be. (i.e., OK for grammatically correct MS programs and ERROR otherwise).

- No homework will be accepted if it is not submitted using SUCourse.
- You may get help from our TAs, LA, or friends. However, **you must write your bison 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 cs305.sabanciuniv.edu), there can be incompatibilities once you transfer them to the cs305 machine. Since the grading will be done automatically on the cs305 machine, we strongly encourage you to do your development on the cs305 machine, or at least test your code on the cs305 machine before submitting it. If you prefer not to test your implementation on the cs305 machine, this means you accept to take the risks of incompatibilities. Even if you have spent hours on 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 minutes of delay.
- We will not accept any homework later than 500 minutes after the deadline.
- SUCourse's timestamp will be used for STF computation.
- If you submit multiple times, the last submitted version and its submission time will be used.