

Assignment Report

Telecommunication Software
Lab

ELP 718

Name: Anupam Kumar Jha

Entry number: 2016JTM2087

Assignment no. 12

Submission Date: November 3, 2016



*Indian Institute of Technology
Delhi*

November 3, 2016

Contents

1	Introduction	3
2	Problem Statements	4
2.1	Problem Statement 1	4
2.2	Problem Statement 2	4
3	Assumptions	5
3.1	Assumptions for Problem Statement 1	5
3.2	Assumptions for Problem Statement 2	5
4	Logic and Impletantaion	6
4.1	Implementation for Problem Statement 1	6
4.2	Implementation for Problem Statement 2	6
5	Test Description and Results	7
5.1	Tests for Problem Statement 1	7
5.2	Tests for Problem Statement 2	7
6	Screenshots	8
7	References and citations	12
8	Epilogue	13

List of Figures

1	Screenshot1 for problem statement 1	8
2	Screenshot2 for problem statement 1	9
3	Screenshot2 for problem statement 1	10
4	Screenshot2 for problem statement 1	11

1 Introduction

Lex is officially known as a "Lexical Analyser".

It's main job is to break up an input stream into more usable elements. Or in, other words, to identify the "interesting bits" in a text file.

For example, if you are writing a compiler for the C programming language, the symbols `{}` , `()` ; all have significance on their own. The letter `a` usually appears as part of a keyword or variable name, and is not interesting on it's own. Instead, we are interested in the whole word. Spaces and new-lines are completely uninteresting, and we want to ignore them completely, unless they appear within quotes "like this".

All of these things are handled by the Lexical Analyser.

Yacc is officially known as a "parser". It's job is to analyse the structure of the input stream, and operate of the "big picture". In the course of it's normal work, the parser also verifies that the input is syntactically sound. Consider again the example of a C-compiler. In the C-language, a word can be a function name or a variable, depending on whether it is followed by a `(` or a `=`. There should be exactly one `}` for each `{` in the program.

YACC stands for "Yet Another Compiler Compiler". This is because this kind of analysis of text files is normally associated with writing compilers.

2 Problem Statements

2.1 Problem Statement 1

We have to write a lex/yacc code to evaluate the consistency of given .pgn file, which record chess moves in given game. You have to find any wrong notations and its location in metadata or using move number. The PGN file follows international notation rules. THE PGN file has metadat and record of pawn moves.

2.2 Problem Statement 2

We are given list of students in given format with their marks in respective subjects. We have to calculate their SGPA and write that down after every student. Input file has entry number, name of student, followed by marks in respective subjects.

3 Assumptions

3.1 Assumptions for Problem Statement 1

Assumptions for problem statement 1 are:

- Assumption no 1 : The matadata of PGN file has only seven fields.
- Assumption no 2 : The chess moves are not to be validated. Only the syntax of the recorded chess moves is to be validated.

3.2 Assumptions for Problem Statement 2

Assumptions for problem statement 2 are:

- Assumption no 1 : SGPA is calcualted using IIT Delhi rules.
- Assumption no 2 : Dropped and Audit course are not taken into account while calculating SGPA.

4 Logic and Impletantaion

4.1 Implementation for Problem Statement 1

- Write regular expressions to check the metadata. There are seven fields and specific input formats to check.
- Write regular expression for chess moves. Consider only normal moves where no peice is killed.
- Consider case where some piece is killed. Write regex for that.
- Declare variables val and inval, mvno, nmeta, count. Initialize all to zero.
- Increment mvno whenever regex for valid index is detected. Increment val whenever valid pattern is detected. IF mvno and val are not equal, error has occured.
- Similarly, nmeta is incremented whenever valid meta data is detected. At the end, if nmeta is less than 7, there is error in meta data.

4.2 Implementation for Problem Statement 2

- Write regular expressions to check the input file format.
- When the format is proper, copy the line to output file.
- Extract credits obtained and the weightage of each subject using appropriate field position. Do weighted sum and divide by no of subjects to get the SGPA.

5 Test Description and Results

5.1 Tests for Problem Statement 1

The code runs properly as asked for:

- When a standard file is given in input, no error is displayed.
- Introduce an error at move no 20 by giving a specail symbol. The output will show invalid data at move 20.
- Similarly, introduce a format error at any other move no. Error is displayed for that perticular move no.
- If admin is logged in, Marks button displays students marks graph.

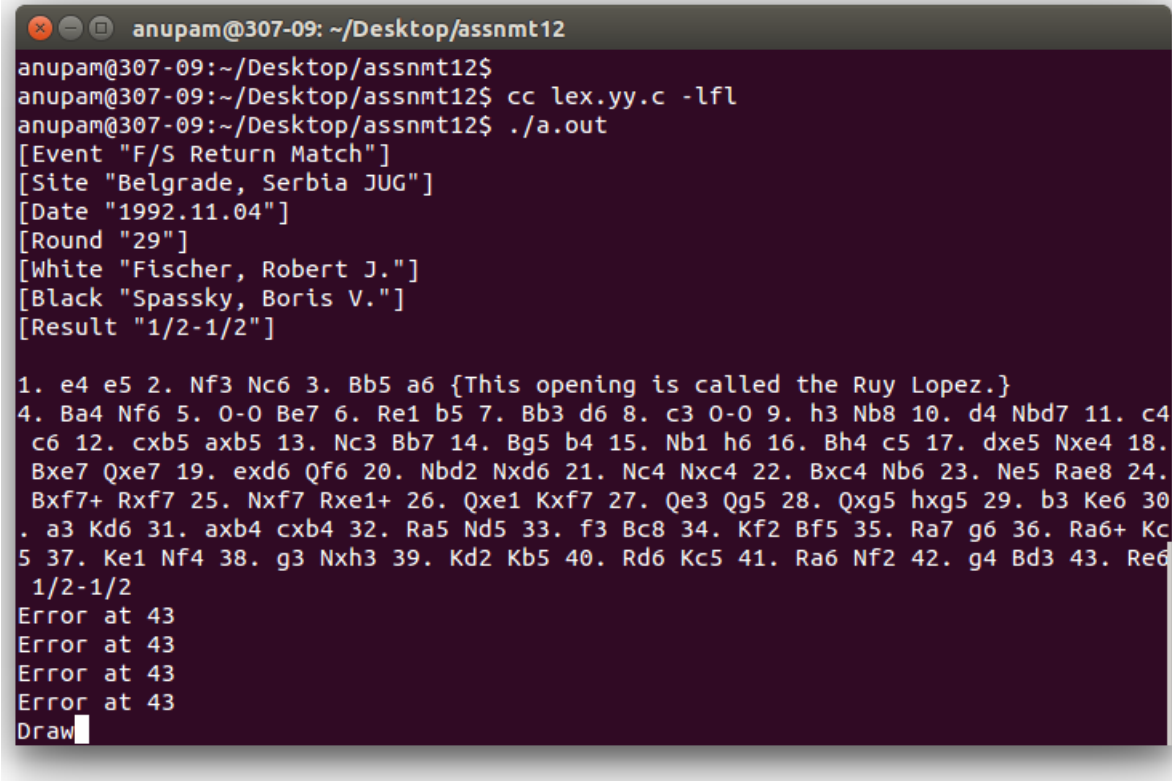
5.2 Tests for Problem Statement 2

The code runs properly as asked for:

- Give entry no with invalid year. It displays error.
- Give a literal in place of number for grade. Error is displayed.
- Give 3 students' data with 2 or more subjects. SGPA is properly displayed.
- Use Dropped and audit courses. The output does not count them for calculating SGPA.

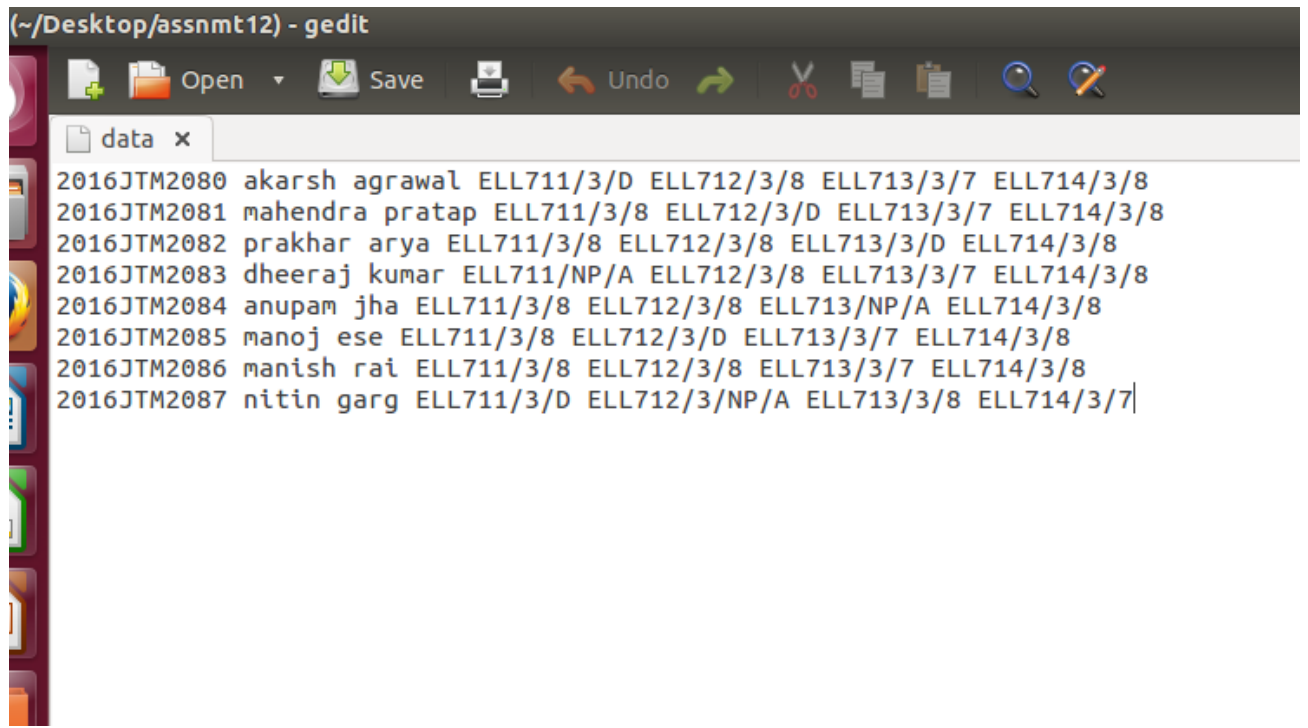
6 Screenshots

Following are the screenshots for program output:

A screenshot of a terminal window with a dark purple background and light-colored text. The window title is 'anupam@307-09: ~/Desktop/assnmt12'. The terminal shows the execution of a program that outputs chess game data. The output includes event information, site, date, round, player names, and a list of 43 moves. The game ends with a draw.

```
anupam@307-09:~/Desktop/assnmt12$  
anupam@307-09:~/Desktop/assnmt12$ cc lex.yy.c -lfl  
anupam@307-09:~/Desktop/assnmt12$ ./a.out  
[Event "F/S Return Match"]  
[Site "Belgrade, Serbia JUG"]  
[Date "1992.11.04"]  
[Round "29"]  
[White "Fischer, Robert J."]  
[Black "Spassky, Boris V."]  
[Result "1/2-1/2"]  
  
1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}  
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8 10. d4 Nbd7 11. c4  
c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5 Nxe4 18.  
Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6 23. Ne5 Rae8 24.  
Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5 hxg5 29. b3 Ke6 30  
. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5 35. Ra7 g6 36. Ra6+ Kc  
5 37. Ke1 Nf4 38. g3 Nxe3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6 Nf2 42. g4 Bd3 43. Red  
1/2-1/2  
Error at 43  
Error at 43  
Error at 43  
Error at 43  
Draw
```

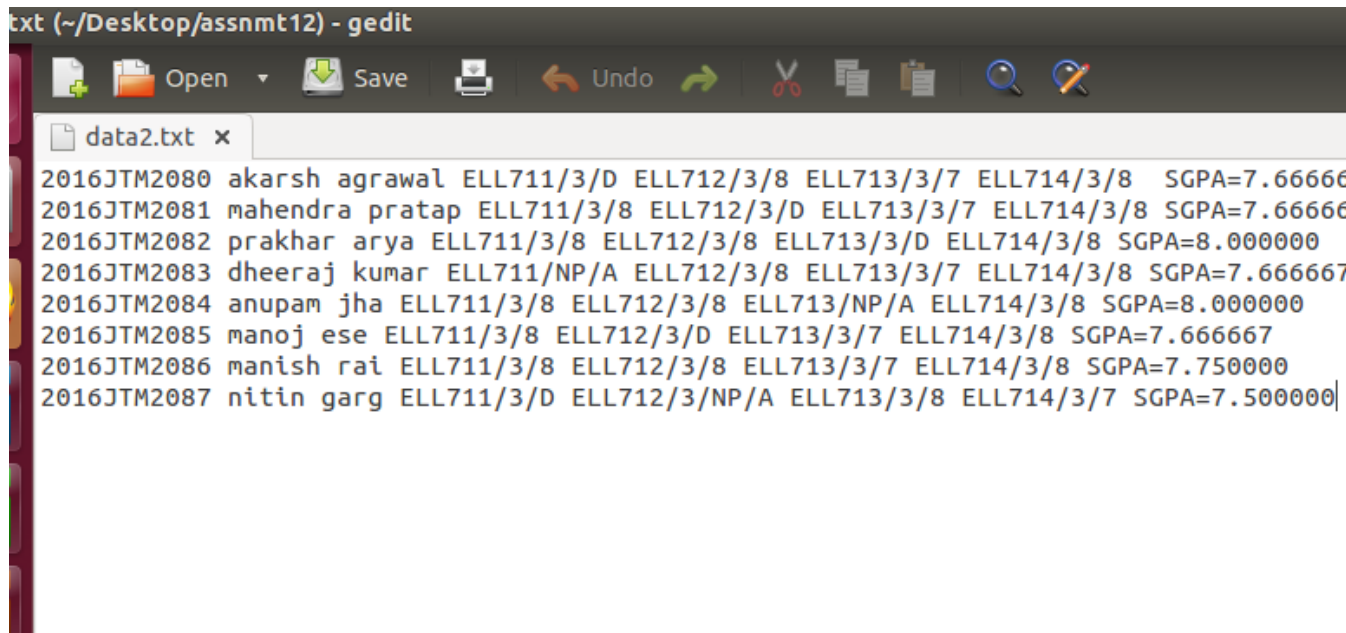
Figure 1: Screenshot1 for problem statement 1



The screenshot shows a gedit text editor window titled "(~/Desktop/assnmt12) - gedit". The window contains a single file named "data x" with the following text:

```
2016JTM2080 akarsh agrawal ELL711/3/D ELL712/3/8 ELL713/3/7 ELL714/3/8
2016JTM2081 mahendra pratap ELL711/3/8 ELL712/3/D ELL713/3/7 ELL714/3/8
2016JTM2082 prakhar arya ELL711/3/8 ELL712/3/8 ELL713/3/D ELL714/3/8
2016JTM2083 dheeraj kumar ELL711/NP/A ELL712/3/8 ELL713/3/7 ELL714/3/8
2016JTM2084 anupam jha ELL711/3/8 ELL712/3/8 ELL713/NP/A ELL714/3/8
2016JTM2085 manoj ese ELL711/3/8 ELL712/3/D ELL713/3/7 ELL714/3/8
2016JTM2086 manish rai ELL711/3/8 ELL712/3/8 ELL713/3/7 ELL714/3/8
2016JTM2087 nitin garg ELL711/3/D ELL712/3/NP/A ELL713/3/8 ELL714/3/7|
```

Figure 2: Screenshot2 for problem statement 2 (1st fig)



The screenshot shows a gedit text editor window titled "data2.txt (~/Desktop/assnmt12) - gedit". The window contains a list of student records, each with a unique ID, a name, and a series of course grades followed by the SGPA. The records are as follows:

ID	Name	ELL711/3/D	ELL712/3/8	ELL713/3/7	ELL714/3/8	SGPA
2016JTM2080	akarsh agrawal					7.66666
2016JTM2081	mahendra pratap					7.66666
2016JTM2082	prakhar arya					8.000000
2016JTM2083	dheeraj kumar					7.666667
2016JTM2084	anupam jha					8.000000
2016JTM2085	manoj ese					7.666667
2016JTM2086	manish rai					7.750000
2016JTM2087	nitin garg					7.500000

Figure 3: Screenshot2 for problem statement 2 (2nd fig)

```
anupam@307-09: ~/Desktop/assnmt12
flex__bison.pdf      PS1.png             yacc-lex video lectures
flowchart practice.png ps1.py              yacc ppt.pdf
garcia pprob _files  ps21.png
anupam@307-09:~/Desktop$ cd assmt 12
bash: cd: assmt: No such file or directory
anupam@307-09:~/Desktop$ cd assmnt 12
bash: cd: assmnt: No such file or directory
anupam@307-09:~/Desktop$ cd assnmt 12
bash: cd: assnmt: No such file or directory
anupam@307-09:~/Desktop$ cd assnmt12
anupam@307-09:~/Desktop/assnmt12$ flex ps2.l
anupam@307-09:~/Desktop/assnmt12$ flex ps2.l
anupam@307-09:~/Desktop/assnmt12$ flex ps2.l
anupam@307-09:~/Desktop/assnmt12$ cc lex.yy.c -lfl
anupam@307-09:~/Desktop/assnmt12$ ./a.out data data2.txt
Avg is when new line is detected 7.666667
Avg is when new line is detected 7.666667
Avg is when new line is detected 8.000000
Avg is when new line is detected 7.666667
Avg is when new line is detected 8.000000
Avg is when new line is detected 7.666667
Avg is when new line is detected 7.750000
Avg is when new line is detected 7.500000
anupam@307-09:~/Desktop/assnmt12$
```

Figure 4: Screenshot2 for problem statement 2 (3rd fig)

7 References and citations

References

- [1] Online reference for LEX and YACC. Available:
<http://dinosaur.compilertools.net/>
- [2] LEX and YACC tutorial. Availabe: <http://epaperpress.com/lexandyacc/download/LexAndYacc.pdf>
- [3] Basic introduction to LEX and YACC. Available:
https://luv.asn.au/overheads/lex_yacc/

8 Epilogue

This assignment has taught us use of LEX and YACC for basic text processing.