Module 1a (10 marks)
Design and Implement Lexical Analyzer

Design and implement a lexical analyzer from scratch whose specification are given below, by using your favorite programming language (preferably C). The lexical analyzer identifies and outputs tokens (valid words and punctuations) in the source program.

id = letter alphanum*
alphanum = letter | digit | _
num = integer | float
integer = nonzero digit* | 0
float = integer fraction
fraction = .digit* nonzero | .0
letter = a..z |A..Z
digit = 0..9
nonzero = 1..9

Operators, punctuation and reserved words

==	+	(	if	void
<>	-	)	then	main
<	*	{	else	
>	/	}	for	
<=	=	[	class	
>=	and	]	int	
;	not	/*	float	
,	or	*/	get	
.	//	put	return	

If a token belongs to both category, the code should output the one that is written higher (5 should be identified as alphanum, not integer; a should be identified as id, not alphanum)

Your code should ignore whatever is written inside the comment block (whatever is written inside the /* */ block)

If no match is found, report it as an error token.

If you need to assume something, assume it but with correct justification. Any assumption should not result in diminishing the expressive power of the language.

Sample Input:

```
void main( )
        {
          int a=15, b_, c=5;
          float d=2.2;
          if (c%2==0)    /* checking if c is even */
                        {
                          printf(“a”);
                        }
        }
```

Sample Output:

Operators, punctuations, and reserved words:

void main ( ) { int , , = ; float = ;
if (==) { ( ) ; } }

id:

a b_ c d

alphanum:

5 2 0

num:

15 2.2

error token:

% printf “ ”

Module 1b (10 marks)
Chomsky Hierarchy Presentation

Study deeply about Chomsky Hierarchy and prepare a powerpoint presentation. Your presentation should address the following:

- What is Chomsky Hierarchy, when was it proposed, what was the background etc.
- The basic diagram for Chomsky Hierarchy with detailed example of each of the four classes
- A description of the automata that is used for recognizing each class
- An explanation on how each class is a proper subset of the classes above it

For making this presentation, you might need to study some advanced topics such as pushdown automata or Turing Machine, which have not been covered in the class. It's your responsibility to
understand these topics on your own based on the knowledge you obtained in the class. You should also be able to make others understand these topics.

The duration of the presentation will be minimum 15 minutes, maximum 18 minutes.

Although the duration of the presentation is 15-18 minutes, your slides should address all of the topics mentioned above. Based on the available time, the instructor might ask you to skip some parts of the slide.

Module 2 (5 marks)
Leftmost and Rightmost Traversal Visualization

A leftmost traversal is a traversal in which we always replace the leftmost nonterminal. On the other hand, a rightmost traversal is a traversal in which we always replace the rightmost nonterminal.

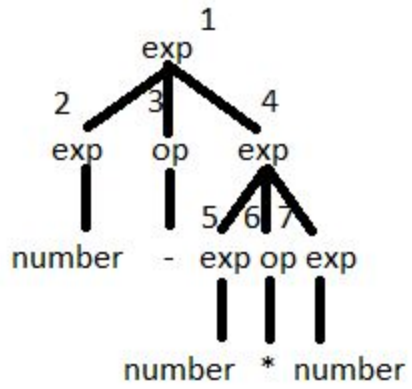
In this assignment, you need to consider the following grammar:
exp -> exp op exp | (exp) | number
op -> + | - | *

You need to write a piece of code that is capable of generating a parse tree from a leftmost derivation of this grammar. You should number the internal nodes of the tree in the order the rules are applied . Finally, you need to traverse the internal nodes of the tree in preorder and verify that both the numberings match.

Next, you need to repeat the same thing for rightmost derivation. In this case, you should verify that the rightmost derivation corresponds to reverse postorder traversal of the tree.

Sample Input (leftmost derivation):
exp => exp op exp
=> number op exp
=> number - exp
=> number - exp op exp
=> number - number op exp
=> number - number * exp
=> number - number * number

Sample Output:
Your code should be able to generate the following numbered parse tree from this derivation:



Next, you should apply the preorder traversal algorithm on this tree and get the following sequence of internal nodes:

exp exp op exp exp op exp

Both of the sequences should match (the one in the tree and the one in the traversal)

Module 3 (5 marks)
Lex Presentation

Examine the lexical analysis tool Lex and prepare a powerpoint presentation. Your presentation should address the following:

- What is Lex, it's short history
- What is the theoretical idea of Lex, and what are the limitations of Lex
- Explanation of a few rules in Lex
- A demo on how Lex could be used to automate the lexical analysis process

Choose the most advanced available version of Lex. After seeing your presentation, students should be able to get a high-level overview on Lex and how it could be used. You should also prepare yourself to answer any question from the audience.

The duration of the presentation will be minimum 10 minutes, maximum 12 minutes.

Although the duration of the presentation is 10-12 minutes, your slides should address all of the topics mentioned above. Based on the available time, the instructor might ask you to skip some parts of the slide.

Module 4 (5 marks)
Yacc Presentation

Examine the parsing tool Yacc and prepare a powerpoint presentation. Your presentation should address the following:

- What is Yacc, it's short history
- What is the theoretical idea of Yacc, and what are the limitations of Yacc
- Explanation of a few rules in Yacc
- A demo on how Yacc could be used to automate the parsing process

Choose the most advanced available version of Yacc. After seeing your presentation, students should be able to get a high-level overview on Yacc and how it could be used. You should also prepare yourself to answer any question from the audience.

The duration of the presentation will be minimum 10 minutes, maximum 12 minutes.

Although the duration of the presentation is 10-12 minutes, your slides should address all of the topics mentioned above. Based on the available time, the instructor might ask you to skip some parts of the slide.