Middle East Technical University **Department of Computer Engineering**

CENG444 - Language Processors

2024-2025 Spring
Project I
Lexical analysis with flex

1. Problem

In this assignment, you are expected to write a program that uses a lexer to tokenize and process some UTF8 encoded text files of a given format. You are required to use `C++` and `flex` to write your program. Your program is expected to read a line from the input file at each step and identify the type of the line, keep and the report the properties as requested.

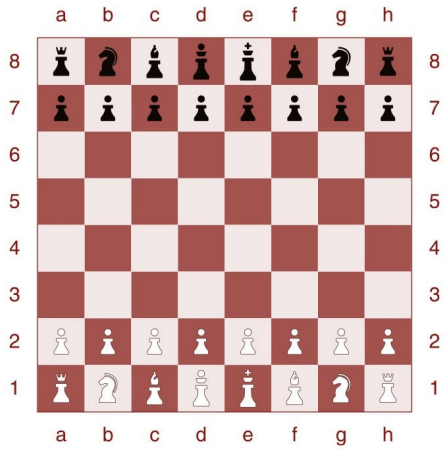
Your program will process a given input file line by line to identify the type of each line with help of the lexical analyzer that you will develop. The primary result of the identification process for each line will be of one of the four categories listed below:

1. The line is in Chess Algebraic Notation.
2. The line contains an IBAN.
3. The line contains an IPv6 address.
4. The line type is unresolved.

Chess Algebraic Notation

You will **not** need a profound knowledge on chess to solve this problem! Here are a few introductory knowledge items that should suffice.

- Chess is an ancient game played by two players with two sets (white and black) of pieces placed in an initial configuration on an 8 by 8 checker board. The rows of board (ranks) are enumerated bottom up ranging 1 to 8. The columns (files) are encoded by letters from a to h.



- One player owns the white (the white player) pieces and other one (the black player) owns the black pieces. The white player starts the game. Players make move in turns. In each turn, a player moves one of his/her/their pieces to a blank square or a square occupied by an opponent piece. In the latter case, the move is referred to as a capture. The ultimate goal of the game is to capture the opponent king.
- There are a bunch of definitions and rules that govern the moves, captures and cases, which are totally irrelevant to this assignment. You are required to identify the notation that encodes moves, captures, and some game specific cases like promotions, checks, and checkmates at an abstract level.

Your program will recognize a line that contains a move (white's move) or a couple of moves (white's move and black's counter-move) that follow a turn number. Encoding of a move is referred to as algebraic notation that can be found in PGN (Portable Game Notation) for chess [1]. Below are the syntactic rules of the chess algebraic notation line that will be applied in this experiment.

1. The line starts with a positive integer followed by a dot and a space. Leading zeros in the integer are not allowed.
2. The line continues with a move (white's) move. Optionally, black's counter-move is specified after a single white space.

   2.1. Regular Move

      2.1.1. If the move is by a king, queen, bishop, knight, or rook then the piece letter appears as the first symbol of the move. See the table below for the symbols, which are all capital letters. If the move is by a pawn, no symbol is used except for the cases the pawn captures an opponent piece. In such a case, the file of the pawn is used.

Piece	Symbol
King	K
Queen	Q
Bishop	B
Knight	N
Rook	R

      2.1.2. On rare occasions, for the pieces other than pawn, the file or the rank of the moving piece is specified to eliminate ambiguities.

      2.1.3. If the move captures an opponent piece an x is placed.

      2.1.4. The destination square of the move is given as file followed by rank.

      2.1.5. When a pawn on a file reaches the final rank, it is promoted. The promoted symbol of the piece that the pawn is promoted to is appended to the move.

   2.2. Castling

      2.2.1. Castling with king-side rook is noted as O-O, castling with queen-side rook is noted as O-O-O. Both zeros or capital O's can be used. But zeros and capital O's cannot be mixed while denoting a castling (i.e. it is all O's or all zeros for an instance).

3. Two optional symbols can be added to the end of the move. + denotes "check!", # denotes checkmate.
4. Files (columns) are specified as a, b, c, d, e, f, g, and h. The files are always in small letters and the pieces are in capital letters to eliminate ambiguity.

IBAN (International Bank Account Number)

International Bank Account Numbers have a simple format that can easily be implemented by using a lexer. You are not required to check the validity of IBAN. You will check syntactic conformance only.

[1] Further details can be found on https://en.wikipedia.org/wiki/Algebraic_notation_(chess), https://www.chess.com/terms/chess-notation . Note that, the rules that will be applied in this experiment are slightly reduced subset of the rules found in the pages referred to.

An IBAN starts with a two-letter country code[2]. The country code is followed by up to 30 alphanumeric characters. You are required to implement the syntactic check for each country listed in the table below [3]. The code and the length fields must be strictly conformant.

Country	Code	Length	IBAN Example
Albania	AL	28	AL35202111090000000001234567
Andorra	AD	24	AD1400080001001234567890
Austria	AT	20	AT483200000012345864
Azerbaijan	AZ	28	AZ77VTBA00000000001234567890
Bahrain	BH	22	BH02CITI00001077181611
Belarus	BY	28	BY86AKBB10100000002966000000
Belgium	BE	16	BE71096123456769
Bosnia and Herzegovina	BA	20	BA393385804800211234
Brazil	BR	29	BR1500000000000010932840814P2
Bulgaria	BG	22	BG18RZBB91550123456789
Burundi	BI	27	BI13200011000100000123456789
Costa Rica	CR	22	CR23015108410026012345
Croatia	HR	21	HR1723600001101234565
Cyprus	CY	28	CY21002001950000357001234567
Czech Republic	CZ	24	CZ5508000000001234567899
Denmark	DK	18	DK9520000123456789
Djibouti	DJ	27	DJ21100020100104099430200008
Dominican Republic	DO	28	DO22ACAU00000000000123456789
Egypt	EG	29	EG800002000156789012345180002
El Salvador	SV	28	SV43ACAT00000000000000123123
Estonia	EE	20	EE471000001020145685
Falkland Islands	FK	18	FK12SC987654321098
Faroe Islands	FO	18	FO9264600123456789
Finland	FI	18	FI1410093000123458
France	FR	27	FR7630006000011234567890189
Georgia	GE	22	GE60NB0000000123456789
Germany	DE	22	DE75512108001245126199
Gibraltar	GI	23	GI56XAPO000001234567890
Greece	GR	27	GR9608100010000001234567890
Greenland	GL	18	GL8964710123456789
Guatemala	GT	28	GT20AGRO00000000001234567890
Holy See (the)	VA	22	VA59001123000012345678
Honduras	HN	28	HN54PISA00000000000000123124
Hungary	HU	28	HU93116000060000000012345676
Iceland	IS	26	IS750001121234563108962099
Iraq	IQ	23	IQ20CBIQ861800101010500
Ireland	IE	22	IE64IRCE92050112345678
Israel	IL	23	IL170108000000012612345
Italy	IT	27	IT60X0542811101000000123456
Jordan	JO	30	JO71CBJO0000000000001234567890
Kazakhstan	KZ	20	KZ244350000012344567
Kosovo	XK	20	XK051212012345678906
Kuwait	KW	30	KW81CBKU0000000000001234560101

[2] These codes are known as ISO-3166 (alpha-2) codes.
[3] Table is reduced from the original that can be found in https://www.iban.com/structure .

Country	Code	Length	IBAN Example
Latvia	LV	21	LV97HABA0012345678910
Lebanon	LB	28	LB92000700000000123123456123
Libya	LY	25	LY3802100100000123456789
Liechtenstein	LI	21	LI7408806123456789012
Lithuania	LT	20	LT601010012345678901
Luxembourg	LU	20	LU120010001234567891
Malta	MT	31	MT31MALT01100000000000000000123
Mauritania	MR	27	MR1300020001010000123456753
Mauritius	MU	30	MU43BOMM0101123456789101000MUR
Moldova	MD	24	MD21EX000000000001234567
Monaco	MC	27	MC5810096180790123456789085
Mongolia	MN	20	MN580050099123456789
Montenegro	ME	22	ME25505000012345678951
Netherlands	NL	18	NL02ABNA0123456789
Nicaragua	NI	28	NI79BAMC0000000000003123123
North Macedonia	MK	19	MK07200002785123453
Norway	NO	15	NO8330001234567
Pakistan	PK	24	PK36SCBL0000001123456702
Palestine	PS	29	PS92PALS000000000400123456702
Poland	PL	28	PL10105000997603123456789123
Portugal	PT	25	PT5000270000001234567833
Qatar	QA	29	QA54QNBA000000000000693123456
Romania	RO	24	RO66BACX0000001234567890
Russia	RU	33	RU0204452560040702810412345678901
Saint Lucia	LC	32	LC14BOSL12345678901234567890123 4
San Marino	SM	27	SM76P0854009812123456789123
Sao Tome and Principe	ST	25	ST2300020000289355710148
Saudi Arabia	SA	24	SA4420000001234567891234
Serbia	RS	22	RS35105008123123123173
Seychelles	SC	31	SC74MCBL0103123456789012345 6USD
Slovakia	SK	24	SK8975000000000012345671
Slovenia	SI	19	SI56192001234567892
Somalia	SO	23	SO06100000 1123123456789
Spain	ES	24	ES7921000813610123456789
Sudan	SD	18	SD8811123456789012
Sultanate of Oman	OM	23	OM040280000012345678901
Sweden	SE	24	SE7280000810340009783242
Switzerland	CH	21	CH5604835012345678009
Timor-Leste	TL	23	TL380010012345678910106
Tunisia	TN	24	TN5904018104004942712345
Turkey	TR	26	TR320010009999901234567890
Ukraine	UA	29	UA903052992990004149123456789
United Arab Emirates	AE	23	AE460090000000123456789
United Kingdom	GB	22	GB33BUKB20201555555555
Virgin Islands, British	VG	24	VG07ABVI0000000123456789
Yemen	Ye	30	YE09CBKU00000000000001234560101

Important Note: Only the English capital letters are allowed. Your program must not recognize small letters as valid syntactical entities. Process the data given in the table and produce a long, complex expression to create the lexer. The table is also given as a companion file. Even if more efficient, any technique utilizing other than sole lexer rules will not be accepted!

IPv6 Addresses

You are required to implement a lexical analyzer to recognize a limited subset of IPv6 address notation. The size and format of the IPv6 address is defined as follows[4] .

1. The IPv6 address size is 128 bits. The preferred IPv6 address representation is: x:x:x:x:x:x:x:x, where each x is the hexadecimal values of the eight 16-bit pieces of the address. IPv6 addresses range from 0000:0000:0000:0000:0000:0000:0000:0000 to ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff.
2. In addition to this preferred format, IPv6 addresses might be specified in two other shortened formats:

   2.1. Omit leading zeros

   Specify IPv6 addresses by omitting leading zeros. For example, IPv6 address 1050:0000:0000:0000:0005:0600:300c:326b can be written as 1050:0:0:0:5:600:300c:326b.

   2.2. Double colon

   Specify IPv6 addresses by using double colons (::) in place of a series of zeros. For example, IPv6 address ff06:0:0:0:0:0:0:c3 can be written as ff06::c3. Double colons can be used only once in an IP address.

The IPv6 address specification has much more as per syntax and semantics such as CIDR notation, address types and ranges. The rules noted above are enough for the purposes of this experiment.

2. Your Assignment

You will develop a C++ program that accepts one command line argument as the text file which will be processed line by line. Your program will generate an output file that contains each line found in the input file. Each line will be appended by " => " and line identification. The name of the output file will be based on the name of the input file. The extension of the input and the output files will be assumed as txt, so the input file name will be specified without extension. The input parameter <p> will imply the input file <p>.txt. The name of the output file will be generated by appending "-out" to the input file name. So, the output file name will be <p>-out.txt.

Example run:

When your program is compiled and run as "project1 test1", your program will read the input file "text1.txt", and generate output as "test1-out.txt". In case your program fails to open the input file, your program will print "File not found!" error message and terminate. An example of respective input and output file lines is given below.

[4] More can be found in https://www.ibm.com/docs/en/i/7.4?topic=concepts-ipv6-address-formats

Input (test1.txt)	Output (test1-out.txt)
1. e4	1. e4 => Chess Algebraic Notation
PL10105000997603123456789123	PL10105000997603123456789123 => IBAN Number
PL10105000997603123456789123 4	PL10105000997603123456789123 4 => Unresolved
2001:db8::	2001:db8:: => IPv6
fe80::1ff:fe23:4567:890a	fe80::1ff:fe23:4567:890a => IPV6
fe80::1ff::fe23:4567:890a	fe80::1ff::fe23:4567:890a => Unresolved
22. Qxf4 Ne2+	22. Qxf4 Ne2+ => Chess Algebraic Notation
23. Qxf4 O-O	23. Qxf4 O-O => Chess Algebraic Notation

In addition to the output file, your program will generate a table of counts to standard output as noted below.

Chess Algebraic Notation: <C1>, where <C1> is the number of Chess Algebraic Notations detected.

IBAN: <C2>, where <C2> is the number of IBANs detected.

IPv6: <C3>, where <C3> is the number of IPv6 addresses detected.

Unresolved: <C4>, where <C4> is the number of unrecognized lines.

Captures: <C5>, where <C5> is the number of captures found in Chess Algebraic Notation Lines

Castling moves: <C6>, where <C6> is the number of castling moves.

Checks: <C7>, where <C7> is the number of checks.

Checkmates: <C8>, where <C8> is the number of checkmates.

Full IPV6: <C9>, where <C9> is the number of full IPv6 addresses. These are the numbers that do not use shortenings.

Example:

Chess Algebraic Notation: 3

IBAN: 1

IPv6: 2

Unresolved: 2

Captures: 1

Castling moves: 1

Checks: 1

Checkmates: 0

Full IPv6: 0

3. Regulations & Hints

- **Implementation:** You should use `flex` with `C++` to write your program. Make sure that your program will accept the name of the input file. In case the program is run without a parameter or with more than one parameter your program must terminate immediately with appropriate error message sent to the standard output.
- **Testing:** Your programs will be tested on an Ubuntu (22.04) machine. It is strongly advised that you set up a virtual machine if you do not have access to a machine with Ubuntu or any other Linux distro.
- **Debugging:** You are advised to use Eclipse IDE for C/C++ for debugging and tracing your program. This will come in handy, especially for the later stages of the project.
- **Evaluation:** The evaluation will be based on correctness of four output items listed below.

Item	Weight
Output File Correctness	7 points for 100% of lines are correct
	5 points for 80% - 99% of lines are correct
	3 points for 50% - 79% of lines are correct
	0 otherwise
Table of Counters Correctness	3 Points for 9 correct lines
	2 points for 6-8 lines
	1 points for 3-5 lines,
	0 otherwise

- **Submission:** You need to submit all relevant files you have implemented (.cpp, .h, .l, etc.) as well as a `README` file with instructions on how to run, in a single `.zip` file named `<your studentID>`.

Following notes are for more clarity to address possible concerns and / or questions.

- Use flex to generate C++ not C file. C implementation will not be accepted.
- Never modify the automatically generated lex.yy.cc file. The evaluation phase use flex as follows and cause regeneration of lexer.

  ```
  flex -+ project01lex.l
  ```

- Subclass the generated lexer class to implement the assignment project. Do not rely on globals to implement state. The only global can be the lexer instance
- Describe precisely the steps to build and run your program in README.txt file, which is essential part of your submission. Provide any configuration items such as scripts, configuration files that will be necessary.
- For your convenience, you are given a companion xlsx file for the IBAN rules. You can process it as you wish while creating the IBAN lexer rule(s).