Programming Hackathon/Exam I
Due Thursday, April 28, 2022 at 11:59 PM

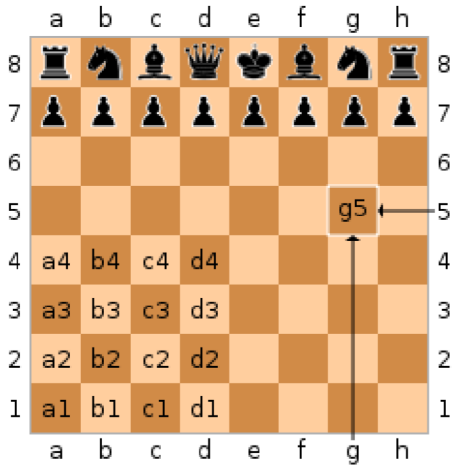
Instructions

- In this assignment, we will introduce you to the problems based upon LEX (Flex).
- You can install Flex, if needed using *apt install* in your Linux machines. For more details, please go through the *man* pages or read online manuals.
- We are providing you with a directory structure with MakeFile.
- Download *CS3320_Lex_Exam.tar.gz*. It has a file hierarchy with Makefiles for you to work on and make submission.
- Your solution should be in the directory corresponding to each question. For example: Lex code for the question P1 should be in p1.l under p1 directory.
- To compile the problem, use *make*. For example: To compile P1, use *make p1* from the outermost directory. Explore Makefile for more details.
- You can safely assume that the inputs are error free; i.e. there is no need to do error handling on the inputs.
- Go through the README file and update it accordingly.
- Provide test cases for each problem.
- Your submission should contain the solutions, test cases and README file in the given directory structure. Compress the outermost directory as *CS3320_Lex_Exam_*<roll-number>.tar.gz and submit the same.
- Don't change the contents of the provided file or directory structure as we would use an automated script to evaluate your submission. *Else, your submission may go unevaluated.*
- Please note that late submissions are not allowed.
- This is an individual assignment.

P1) Let's play Chess!

Given a regular 8 X 8 chessboard in initial state and a set of moves, show the resulting board after making the listed moves, using Lex.

Each tile of the chess board is represented by a coordinate pair consisting of a letter and a number. Columns are identified with letters from *a* through *h* starting from White's left to right. Rows are identified with numbers from *1* to *8* from White's side of the board.



(Source: Wikipedia)

A variation of *Algebraic notation* called, *Long Algebraic Notation* would be used to describe the set of moves. Following are the symbols used in this notation to denote the appropriate pieces in a chess board.

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

Pawns are identified by the absence of upper-case letter.

Moves are represented by the piece's uppercase letter followed by the current position and the destination. For example: *e2-e3* denotes that a pawn moves from *e2* to *e3*; *Ra8-a6* denotes the move of a Rook from *a8* to *a6.*

When a move results in *capture,* instead of *-,* an *x* symbol is used. For example: *Ra6xa5* denotes capture of a piece in *a5* when Rook moves from *a6* to *a5.*

Pawn promotes to Queen / Bishop / Knight / Rook when it moves to the last rank in the enemy line. It is represented by an = symbol followed by the symbol of promotion. For example: *a2-a1=Q* indicates promotion of a Pawn to Queen after making a move from *a2* to *a1*.

Input and Output notation:

Input is the sequence of numbered *turns*, where each turn contains a space separated information on moves made by White followed by Black in that turn.
Output is the resulting chess board in the following format:

	a	b	c	d	e	f	g	h
1	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
2	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
3	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
4	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
5	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
6	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
7	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>
8	<p>	<p>	<p>	<p>	<p>	<p>	<p>	<p>

Where <p> denotes the symbol for each piece. Use R, N, B, K, Q, P for Rook, Knight, Bishop, King, Queen and Pawn respectively. <p> slot may be empty if there are no pieces in that slot. There should be a tab space between each literal in the output and one new line space (i.e. two '\n') between adjacent lines. Note that the white pieces start from *a1* as described earlier. It would be good to assume that the inputs would have moves only in the said format - No checks/checkmates shall be encountered, no castling, etc.

Sample:

Input:

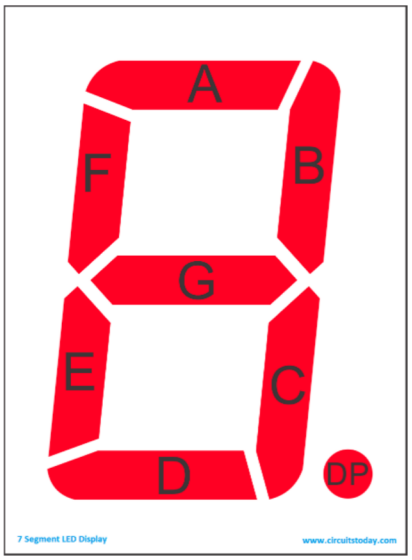
1. e2-e4 e7-e5
2. Ng1-f3 Nb8-c6
3. Bf1-b5 Ng8-f6
4. Bb5xc6 b7xc6

Output:

	a	b	c	d	e	f	g	h
1	R	N	B	Q	K			R
2	P	P	P	P		P	P	P
3						N		
4					P			
5					P			
6			P			N		
7	P		P	P		P	P	P
8	R		B	Q	K	B		R

P2) Seven segment display code generator

A seven segment display consists of 8 LEDs which can be turned on so as to display the desired number. A typical seven segment looks as shown below. Notice the labeling convention for various LEDs.



For displaying 1 we will use following encoding:

```
#     a     b     c     d     e     f     g     dp
      0     1     1     0     0     0     0     0
```

For displaying 1.1 we will use following encoding:

```
#     a     b     c     d     e     f     g     dp
      0     1     1     0     0     0     0     1          →     Seg#1
#     a     b     c     d     e     f     g     dp
      0     1     1     0     0     0     0     0          →     Seg#2
```

This question asks you to write a code generator whose input will be a specification file which describes which LEDs to turn on and output should be C file which we can compile using a sophisticated compiler.

Your program (written in LEX) should take input from the file and generate C code.

Input Notation:

1. Lines starting with # are considered to be comments, and would play no role in the output.
2. Program would start with a call to the custom `INIT` method.
3. `Select` is used for selecting a particular seven segment display in n^{th} position.
4. `Select` is followed by the on/off signals in binary for each component in the selected seven segment display (separated by a tab - \t).
5. Similarly, `Delay` specifies the delay time in seconds.
6. There would be one whitespace after ':' in case of `Delay` and `Select`.
7. The signals can be dependent on a single user input. (Assume that the input would be an integer, read using `readInt` method). Such an input would be provided immediately after `INIT` and before `Select`.
8. `If Else` constructs are allowed.
9. There can be only one seven segment configuration inside if or else blocks, i.e., only one statement can reside in if/else block.
10. If it can take only integer based conditions. Please note that it is sufficient to handle only equality (== and !=), less/greater than and less/greater than equal to conditions.
11. The configuration provided is case insensitive (Both if and IF should be generated as if(). This holds true for any statement in the configuration)

Output notation:

Your output should be a C file with appropriate headers included.

1. The custom functions in the input should be translated to appropriate C style equivalents (as shown in the examples).
2. Assume the definitions of init, select, write, delay are provided in the “seven_segment.h” header file.
3. Assume the displays continue for infinite time(i.e. generate the configuration inside an infinite while loop).

Please check the below examples for more details.

Since it is largely a code-generation problem, we will evaluate your solution based on effective usage of regex or grammar or combination. We will also analyze your code for style, modularity. For sample input we expect the following .c file to get generated.

Sample:

Input file(1)

```
# This specification uses three 7-segment displays in series
# Display value of PI=3.14
INIT
Select: 1
#    a    b    c    d    e    f    g    dp
#Display 3 followed by decimal point
     1    1    1    1    0    0    1    1
Delay: 40
Select: 2
#Display 1
     0    1    1    0    0    0    0    0
Delay: 40
Select: 3
#Display 4
     0    1    1    0    0    1    1    0
Delay: 40
```

Expected Output file:

```
#include<stdio.h>
#include<stdlib.h>
#include<seven_segment.h>
int main()
{
     init();
     while(1)
     {
          select(1);
          write(strtol("11110011"));
          delay(40);
          select(2);
```

```
          write(strtol("01100000"));
          delay(40);
          select(3);
          write(strtol("01100110"));
          delay(40);
     }
}
```

Input file(2)

```
# This specification uses one 7-segment display.
# Depending upon input it prints zero or one.
INIT
input: int
input = readInt
Select: 1
if input == 0:
#    a    b    c    d    e    f    g    dp
#Display 0
     1    1    1    1    1    1    0    0
Else:
#Display 1
     0    1    1    0    0    0    0    0
Delay: 40
```

Expected Output file:

```
#include<stdio.h>
#include<stdlib.h>
#include<seven_segment.h>
int main()
{
     init();
     int input;
     while(1)
     {
          input =  readInt();
          select(1);
          if(input==0)
          {
               write(strtol("11111100"));
```

```
            else
            {
                 write(strtol("01100000"));
            }
            delay(40);
      }
}
```

Anti-Plagiarism Policy followed by the department:

Please go through the plagiarism policy followed by the department (In particular the FAQ section) - https://cse.iith.ac.in/academics/plagiarism-policy.html. Strictly adhere to the same.