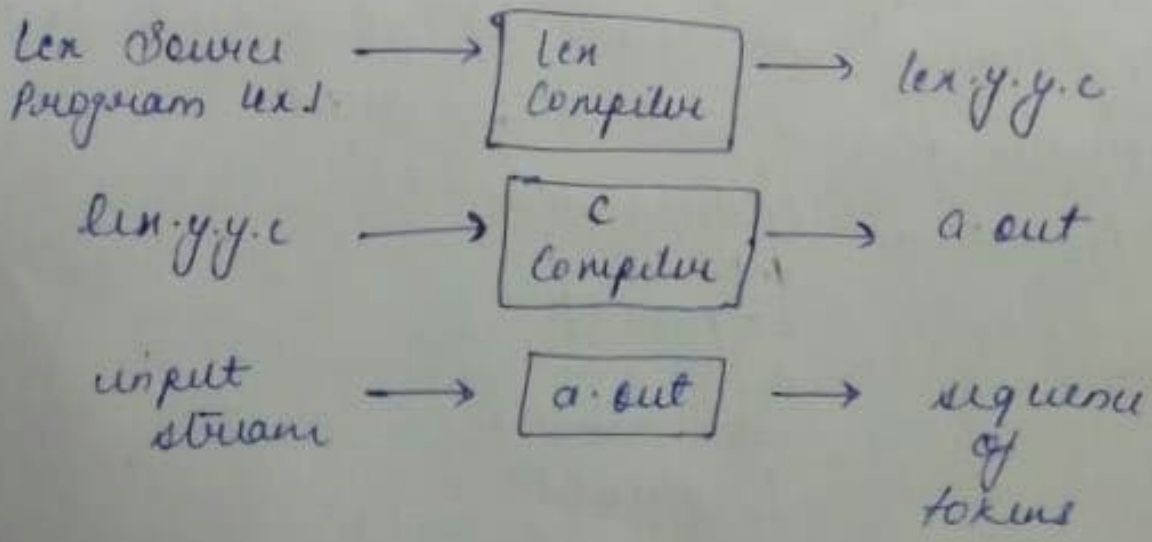
Ques :- What is LEX ? Describe Auxiliary definations and Translation rules for LEX with suitable example

Def :- LEX is a program that generates lexical analyzer. It is used with YACC parser generator. The lexical analyzer is a program that transforms an input stream into a sequence of tokens. It reads the input stream and produces the the source code as output through implementing the lexical analyzer in the C program.

The function of LEX is as follow :-

- First lexical analyzer creates a program lex.l in the Lex language. Then Lex compiler runs the lex.l program and produces a C program lex.y.y.c.
- Finally C compiler runs the lex.y.y.c program and produces an object program a.out.
- a.out is a lexical analyzer that transforms an input stream into a sequence of tokens.



Structure of Lex program :-

Lex program will be in following form.

declarations
%%
translation rule
%%
auxiliary rule

Declarations.

This section includes declaration of variable, constants, and regular definitions.

Translation rules.

It contains regular expressions and code segments.

Form: Pattern {Action}

Pattern is a regular expression or regular definition.

Action refers to segment of code.

Auxilliary functions :-

This section holds additional functions which are used in actions. These functions are compiled seperately and loaded with lexical analyzer.

lexical analysis produced by lex starts its process by reading one character at a time until a valid match for a pattern is found.

Example :-

```
/* declarations */
%%
/* Rules */
%%
int main()
{ yylex();
  return 1;
}
```

The auxiliary declarations and auxiliary functions are copied as such to the lex.yy.c file.

Once the code is written lex.yy.c may be generated using the command lex "filename.l" and compiled as gcc lex.yy.c.