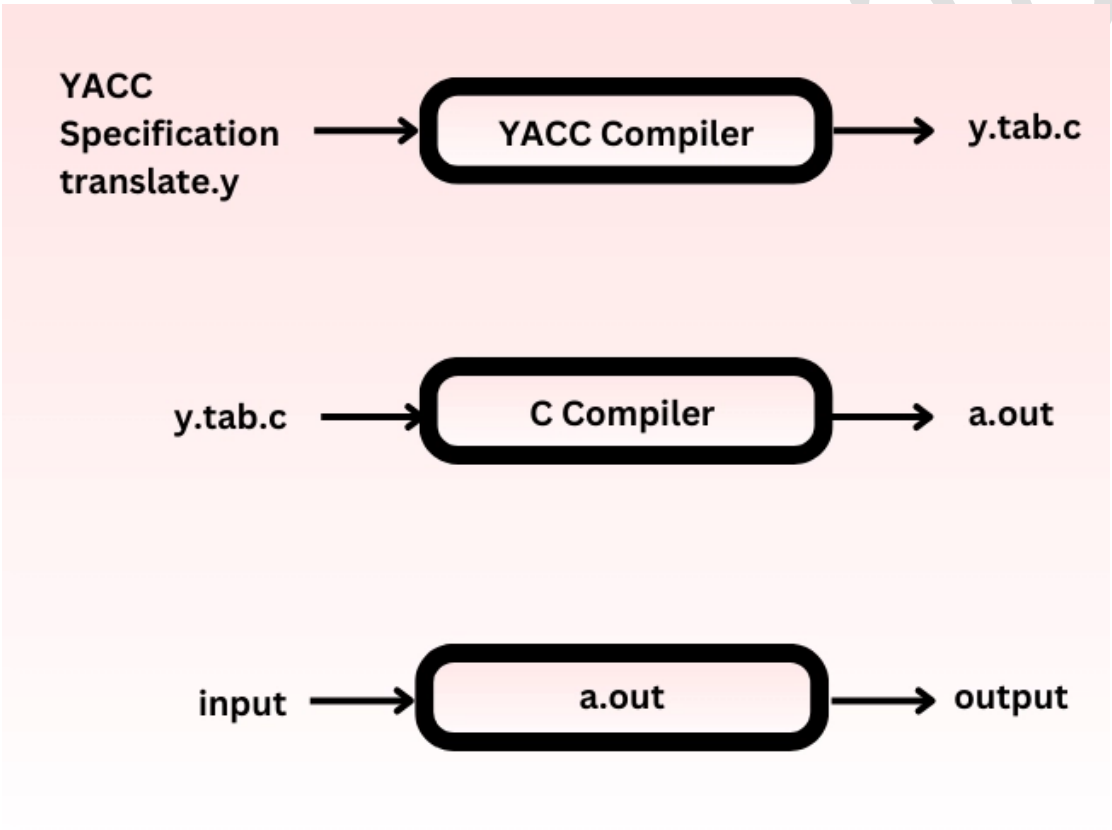
EXP NO : 7 **STUDY OF YACC TOOL**

DATE :

YACC serves as a powerful grammar parser and generator. In essence, it functions as a tool that takes in a grammar specification and transforms it into executable code capable of meticulously structuring input tokens into a coherent syntactic tree, aligning seamlessly with the prescribed grammar rules.

Stephen C. Johnson developed YACC in compiler design in the early 1970s. Initially, YACC was written in the B programming language and was soon rewritten in C. It was originally designed to be complemented by Lex.

In addition, YACC has been rewritten in OCaml, Ratfor, ML, Ada, Pascal, Java, Python, Ruby, and Go.



The input of YACC in compiler design is the rule or grammar, and the output is a C program.

Parts of a YACC Program in Compiler Design

The parts of a YACC program are divided into three sections:

/* definitions */

…….

%%

/* rules */

…….

%%

/* auxiliary routines */

…….

Definition :

These include the header files and any token information used in the syntax. These are located at the top of the input file. Here, the tokens are defined using the %token directive. In YACC, numbers are automatically assigned for tokens.

%token ID

%{ #include <stdio.h> %}

Rules :

The rules are defined between %% and %%. These rules define the actions for when tokens are scanned and are executed when a token matches the grammar.

Auxiliary Routines:

Auxiliary routines contain the functions required in the rules section. This section includes the main() function, where the yyparse() function is always called.

The yyparse() function plays the role of reading tokens, performing actions, and then returning to main() after execution or in case of an error.

It returns 0 after successful parsing and 1 after unsuccessful parsing.

Working of YACC

YACC in compiler design works with the C programming language along with its parser generator:

- An input file with a .y extension is given.
- The file is processed, and two files, y.tab.h and y.tab.c, are created. These files contain code implementing the LALR(1) parser for the grammar.
- This generates a parser that tries to parse valid sentences successfully.
- For the output files:

- If called with the -d option in the command line, YACC produces y.tab.h with all specific definitions.
- If called with the -v option, YACC produces y.output containing a textual description of the LALR(1) parsing table.

How to Execute

1. Save the program code to a file with a .y extension, such as calculator.y.
2. Install Yacc/Bison on your system, if not already installed.
3. Open a terminal or command prompt and navigate to the directory where the .y file is saved.
4. Run the following command to generate a C source file from the Yacc/Bison grammar file:
5. yacc -d filename.y
6. This will create two output files: y.tab.c (the C source file) and y.tab.h (the header file).
7. Compile the generated C source file and link it with the Yacc/Bison runtime library using a C compiler such as GCC:
8. gcc -o calculator y.tab.c -ly
9. Run the compiled program.

Result

The study of the YACC tool has been successfully completed.