**Using Flex and Bison**

1. **Create a Flex File:** Write your lexer specification in a file with a .l extension (e.g., scanner.l). Here’s a simple example:

c

Copy code

%{

#include <stdio.h>

%}

%%

[0-9]+ { printf("Number: %s\n", yytext); }

[a-zA-Z]+ { printf("Word: %s\n", yytext); }

[ \t\n]+ { /\* Ignore whitespace \*/ }

. { printf("Unknown character: %c\n", yytext[0]); }

%%

int yywrap(){

return 1;

}

 **Create a Bison File:** Write your parser specification in a file with a .y extension (e.g., parser.y). Here’s a simple example:

bison

Copy code

%{

#include <stdio.h>

void yyerror(const char \*s);

int yylex(void);

%}

%token NUMBER

%left '+' '-'

%left '\*' '/'

%%

expression:

expression '+' expression { printf("Add\n"); }

| expression '-' expression { printf("Subtract\n"); }

| expression '\*' expression { printf("Multiply\n"); }

| expression '/' expression { printf("Divide\n"); }

| NUMBER { printf("Number: %d\n", $1); }

;

%%

void yyerror(const char \*s) {

fprintf(stderr, "Error: %s\n", s);

}

int main(void) {

yyparse();

return 0;

}

 **Generate the Scanner and Parser Code:** Run Flex and Bison to generate the source files:

flex scanner.l

bison -d parser.y

This will produce lex.yy.c (from Flex) and parser.tab.c and parser.tab.h (from Bison).

 **Compile the Generated Code:** Compile the generated C files using gcc or clang:

gcc lex.yy.c parser.tab.c -o myparser

 **Run Your Program:** Execute the compiled program to test your lexer and parser:

./myparser

You can then provide input through standard input, or redirect input from a file.

flex /home/student/Documents/a.l

[student@localhost ~]$ bison -d /home/student/Documents/parser.y

[student@localhost ~]$ gcc lex.yy.c parser.tab.c -o myparser

[student@localhost ~]$ ./myparser

3+4

Number: 3

Unknown character: +

Number: 4