Students can perform these assignments individually or in a group of 2.

Evaluation: Third week of March i.e March 20 - March 24

Weight: 10 Marks

This is a three part assignment. All assignments have to be done using Lex-Yacc or Flex-Bison.

**Part-I [Any one of the following two] [3 Marks]**

Remove comments from a C program

A "comment" is a sequence of characters beginning with a forward slash/asterisk combination (**/\***) that is treated as a single white-space character by the compiler and is otherwise ignored. A comment can include any combination of characters from the representable character set, including newline characters, but excluding the "end comment" delimiter (**\*/**). Comments can occupy more than one line but cannot be nested.

Comments can appear anywhere a white-space character is allowed. Since the compiler treats a comment as a single white-space character, you cannot include comments within tokens. The compiler ignores the characters in the comment.

Your code should process comments having escaped slash i.e. \/\* is not beginning of a new comment.

OR

Preprocessor for a C program

A C program can have #define. These #define are removed by pre-processing the C code.

Example:

#define PI 3.14

Your program should remove the line having the #define and replace all occurrences of PI in the input C program with 3.14 in the output C program.

Extra credits for making a generic program that removes all #define including multiple dependent #define

#define PI 3.14

#define DOUBLEPI     PI\*2

**Part-II [Any one of the following two] [3 Marks]**

Extend C by defining Complex Number Type

Build a C compiler front-end for processing new type "complex numbers" which will internally represented to be a C structure.

There should be a way to add/subtract/divide and multiply variables of type complex numbers. You should also allow adding of an integer or a real number to a complex number and there should be automatic type coercion. You should create a library "complex.c" and "complex.h" and have definitions of functions: Add(), Subtract(), Multiply(), Divide(), RealPart(), ImaginaryPart(), Equality(), etc.

Add(), Subtract(), Multiply(), Divide(), and Equality(): these functions are called when one of the operands of +, -, \*, /, == is a complex number.

Each occurrence of a complex number should replace it with a structure and each operation should call a function in the library.

OR

Extend C by defining a new power operator

Build a C compiler front-end for processing new operator (^) on integers and floats for raising to power. The operator (^) is right-associative. The operator (^) can be part of mixed expressions having other operators like +, \*, /, -, etc.

You should create a library "power.c" and "power.h" and have definitions of function power(a,b) which takes two arguments a, b and computes a\*a\*a...\*a (b-times).

Each occurrence of operator (^) needs to be replaced with call to power function.

--- Hint ---

One iteration you can process one (^) operator, say you have

a^b^c

then call

power( a, (anything till another operator except ^/ space/ tab/ semicolon/ comma/ question-mark) )

which will result in

power (a, b^c)

and then

power(a, power(b,c))

**Part-III [4 Marks]**

C program having control statements ( if, if-else, switch, while, do-while, for, ?: ) can be written effectively using only if-statement and goto. For example, if-else can be converted into two if-statements, while-statement can be converted to equivalent if(condition) followed by goto, etc.

The C program before and after processing should perform same computation.

You can assume that the statements inside the if-block or while-block are very simple like "a=b;"

Assume that every block starts with '{' and ends with '}'