Phases/Stages of C Compiler:

- ***Editor***
    - C code with .c or .h extension
- ***Preprocessor Stage***
    - Removes comments
    - Expands Macros
    - Output File is : .i
- ***Compiler***
    - Lexical Analyzer
    - Parser
    - Optimizer
    - Code Generator
    - Output File is : .asm(win) or .s (unix)
- ***Assembler***
    - Has binary relocatable modules.
    - Contains machine language but is not in executable format
    - Output files are : .o (unix) / .obj /.rel
- ***Linker (other object modules)***
    - Library (libc.a)
    - Librarian (ar)
    - Has executable as .exe / .com / a.out
    - Specify option -o in unix
- **Compilation of the code is finished**

(Stage-1) About Preprocessor:

- Takes input as .c or .h files
- Two main functions
- Removes all comments
- Processes al the pre processor directives/ expands the macros
- Does not know precedence

(Stage-2) About Compiler:

- Involves 4 phases:
    - Lexical analysis
    - Parsing
    - Optimization
    - Code generation

- The first two phases generates an intermediate code, a code that is similar to assembly language.
- Optimizer takes the intermediate code and tries to make code smaller and faster.
- Code generator translates the output generated from the optimizer stage to assembly language.

(Stage-3) About Assembler:

- Takes input from the compiler stage and generates the output that is close to executable file.
- Assembly files have a common extension in UNIX as .s .
- The output of assembler is and "Relocatable Object Module" a.k.a just Module.
- Translates as much of assembly language into machine language.
- Uses some dummy for placeholders.

- Creates a table to tell the reference of the dummy place holders.
- Modules needs to be patched before it can access an external object(global files, other file locations...etc).

(Stage-4) About Liner or Link Editor:

- Takes the input as the relocatable object modules.
- Replaces the dummy addresses with real addresses by using the table created in the previous stage.
- Files can be linked using the command:
- gcc ld -o storages f1.o f2.o f3.o -lc
- ld creates and executable program called stooges
- lc tells the ld to link the three modules given on the command line with the standard library present at /lib/libc.a.
- Connect all the subroutines and the variables referenced with their declarations.
- A library is a single file that contains all the object modules
- A module is an indivisible chunk of code.

(Stage-5) About Librarians and Libraries:

- The librarians are used to create/maintain the libraries.
- Many linkers scans libraries from front to back as that is the order in which the modules are inserted into the library.
- Run-time library routines do things like double precision arithmetic, process switch statements
- The compiler assumes that a few small subroutines are present in the final program and will generate calls to these routines. This set of routines is called the run-time library.
- Commonly used run-time sub-routines:
    - root or start-up module.
    - root is called from the OS.
    - the main() function is called as a subroutine of root.
    - Root initializes global variables used in I/O library.
    - Assembles argv array from cmd line.
    - Main() is not required if we write our own root module.
- Library Maintenance:
    - Libraries are also called as archives.
    - Archive is a collection of files merged into a single file.
    - Created using librarian.
    - UNIX archive maintenance utility is called ar.
    - Operations on ar: create archive, insert files, delete files, print out the files, replace the arhcive file.
        - syntax: ar rv files.arc f1.c f2.c f3.c f4.c
    - rv tells the archiver what to do. here r means replace the indicated files. here v command tells the archiver to be verbose.
    - to print a file in archived file:
        - ar p files.arc f1.c > ftemp.c (redirect the output to ftemp.c)

During linking, a standard library, libc.a is automatically linked.

If **gcc -c** is used:

The code is compiled but not linked.

If **gcc -Dname** is used along with file name, ex: gcc -DMACROINIFDEF

The code is compiled with the macros enabled.

If **gcc -s** is used:

Generates an assembly source file with the extention .s

If **gcc -o** name is used:

Generates an executable file with specified name instaed of standard a.out.

If **gcc -P** is used:

- Generated .i files or the files generated after the preprocessor stage.