Question 3 - Basic g++ and gcc Compiler Options

C++ source files conventionally use one of the suffixes .C, .cc, .cpp, .CPP, .c+, .cp, or .cxx; C++ header files often use .hh, .hpp, .H, or (for shared template code) .tcc; and preprocessed C++ files use the suffix .ii. GCC recognizes files with these names and compiles them as C++ programs even if you call the compiler the same way as for compiling C programs (usually with the name gcc).
However, the use of gcc does not add the C++ library. g++ is a program that calls GCC and treats .c, .h and .i files as C++ source files instead of C source files unless -x is used, and automatically specifies linking against the C++ library. This program is also useful when precompiling a C header file with a .h extension for use in C++ compilations. On many systems, g++ is also installed with the name c++.

Compilation can involve up to four stages: **preprocessing, compilation proper, assembly and linking**, always in that order. GCC is capable of preprocessing and compiling several files either into several assembler input files, or into one assembler input file; then each assembler input file produces an object file, and linking combines all the object files (those newly compiled, and those specified as input) into an executable file.
When you compile a program as we've been doing, "gcc" checks the source code for errors and creates a binary object file of that code (if no errors exist). It then calls the linker to link your code's object file with other pre-compiled object files residing in libraries. These linked object binaries are saved as your newly compiled program. The options to "gcc" dictate the way in which this process is performed.

Basic compiler options of gcc compiler (-I,-L,-l,-c,-o)

-c {Compile or assemble the source files}: Compile or assemble the source files but do not link. The ultimate output is in the form of an object file for each source file. By default, the object file name for a source file is made by replacing the suffix `.c', `.i', `.s', etc., with `.o'. Unrecognized input files, not requiring compilation or assembly, are ignored.

-o {file_name}: Tells the compiler to save the compiled program under the name . So, typing "gcc myfile.c -o myfile.x" will take the source code of file "myfile.c" and create program "myfile.x" rather than the default program "a.out".

-I {directory_name}: (capital i) Tells gcc that it add the directory to be searched for header files, in addition to the standard headers.

-L {directory_name}: Tells gcc to look in the directory for library files in addition to the standard libraries.

-l {library}: (lower case l) tells the linker to search a standard list of directories for the library, which is actually a file named "lib.a". The linker then uses this file as if it had been specified precisely by name. The directories searched include several standard system directories plus any that you specify with "-L".