Using Make files

See chapter 15 pp 366-369

Some IDE supply tools for maintaining a large multifile program (or project). Usually these are set up in what is called a project. Not all situations afford using such tools which is why the make utility is available in linux environments.

MyList: LLDriver.o LLMergeSort.o myData.o LinkedList.o

gcc -o MyList LLDriver.o LLMergesort.o myData.o LinkedList.o

LLDriver.o: LLDriver.c LinkedList.h LLMergeSort.h myData.h

gcc -c LLDriver.c

LLMergeSort.o: LLMergeSort.c LLMergeSort.h LinkedList.h

gcc -c LLMergeSort.c

myData.o: myData.c myData.h LinkedList.h

gcc -c myData.c

LinkedList.o: LinkedList.c LinkedList.h

gcc -c LinkedList.c

In the example makefile, there are 5 rules.

Rule contains:

1st line: a Target : Files on which the target depends

2nd line: Command to perform to generate the Target if any file on which it  
depends has been modified since the last build

Note: The first line of each rule starts with no spaces/tabs. However, the 2nd line must start after a tab

(spaces do not count).

gcc -c is a switch to ONLY create an object file. Without -c, the target will be an executable.

The creation of the final executable file ALWAYS has to be the first rule.

To run the makefile, you invoke the utility make by type make followed by one of the targets:

make MyList will create the executable MyList.exe

make myData.o will create the myData.o object file

make will always use the first target… The order of the remaining rules does  
 not matter.

Use a text editor to create your makefile. It cannot have a filename extension. Editors such as notepad ALWAYS adds an extension such as .txt. If you have this issue, use the DOS command line utility to rename a file: ren makefile.txt makefile

The equivalent utility in linux is mv (for move file):

mv makefile.txt makefile will do the trick!