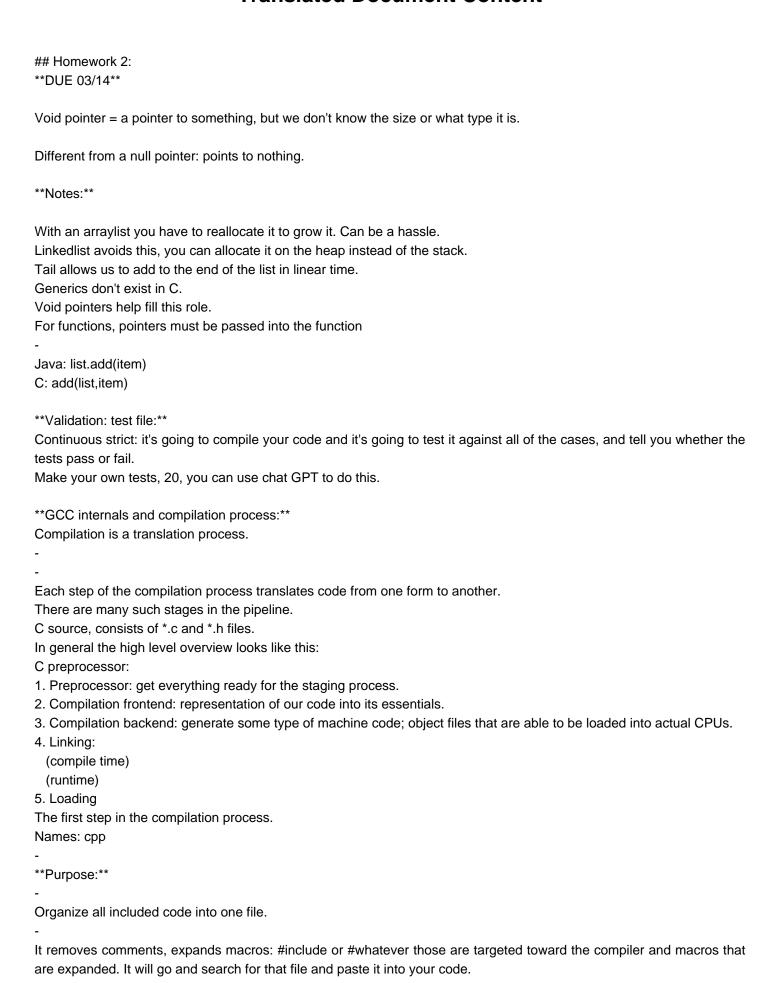
## **Translated Document Content**



#ifndef - if the constant has not been defined.
File type: *.i
**Front end Compilation:**
**Input:** preprocessed C
**Output:** ast
**Program:** cc
**Purpose:**
Check for syntax errors.
Can also check for some semantic errors.
Generate a syntax-independent representation of your code called an:
**ABSTRACT SYNTAX TREE (AST)**
**Program optimization:**
-
-
-
-
After the compilation stage, code exists in tree format.
This tree will be transformed into a number of intermediate representations (IR).
Trees are converted to "generic" (different from Java).
This generic is converted to "gimple".
Gimple is the same code as generic but spaced out rather than condensed.
In a gimple program: you have variable names that can be changed throughout the program: while a SSA form you have
for example: a_1 and a_4 which would be the same variable in a gimple program: a. So SSA you can have different
The purpose of the front/midend IR transformation is to optimize and get the program into the correct backend
representation.
After the backend we can do optimization:
-
-
-
Reduce code by removing redundant calls.
Inline function calls to reduce indirection.
Reorder statements to make them more efficient.
Skip code that produces results that are not consumed.
Remove code from binary that is never called.
Unroll loops.
Automatic tail call optimization:

## **Al Analysis of Legal Terms**

As there is no Spanish text to translate, I cannot fulfill the provided instructions.

Please provide the Spanish text you would like me to translate into English. I will be happy to help once I have the correct source material.