Exercice 1 :

Decrire les etapes de la compilation

1) Analyse lexicale : conversion chaine caractère en symbole (token) il est fait par le lexer (automate issue d'un lexeme) ↓ instancie objet

2) Analyse syntaxique (consomme token) generation arbre syntaxique abstrait (table symbole construite)

3) Analyse semantique

Correction :

COMPILATION
- Traitement par le preprocesseur : gcc -E monprog.c > monpropprep.c (remplace include)
- Compilation proprement dite
  - 1) gcc -S monprog.c → Res = monprog.s
    - 1) analyse et transcription en assembleur
    - 2) assemblage qui genere code obj
  - 2) gcc -c monprog.c / monprog.s] Res = monprog.o
- Edition de lien 3)
  - 3) gcc monprog.o mafonction.o [-o monexe]

Si 3) remarque si on ne precise pas ≠ commande -E, -S ... alors gcc le fait tt seul par defaut

si on l'enlève l'exe devient a.out par defaut et reste ecrasant en exec (pas de pb si qu'un prog dans rep sinon le dernier a.out ecrasera l'ancien)

si le prog est complexe (sch p77)

gcc -c monprog (2) puis (3)

"3) erreur" = undefine reference ou def multiple

TAILLE = nb de chiffre a trouver

car '\0' donc Taille + 1

au minimum 9 octets

pour changer taille block réaligner min (taille_struct) = ?

sizeof (nom) = sizeof(max_type) - sizeof(type_x)

foreach

#pragma pack (?)

↑ padding

On souhaite retourner b = nb chiffre bien placés
m = nb chiffre mal placés

avec m, b ∈ [0 .. 9]

mon_test retourne b * (TAILLE + 1) + m : encodage en base TAILLE + 1 car m̂ si on change la taille ulterieurement

div entiere par T + 1 => b et m = mod T + 1

Taille + 1 valeur car il y a le 0 qui est aussi a encodé
Taille chon + 0

CC : encodera le compilateur que l'on veut utiliser
ainsi en changeant CC on changera tt ses occurences

clean : (cible)
tab → rm *.o mon_exe | nettoie tout les .o car clean renvoie rien

make clean

en haut mk file all : client serveur dernier → ∅ (tab)
↑ cible

si question a man make

dans un m̂ dossier on peut avoir 2 Makefile un avec maj (Makefile) devant (et l'autre : makefile) or l'un est prioritaire sur l'autre.