EX1.1

Avec l’option -fopenmp, il affiche 12 Hello et 1 seul world (12 en default depend du CPU), on peut configure avec

# pragma omp parallel num\_threads(2)

Sans l’option, il affiche un hello et un world.

Ex1.2

printf (" Thread %d: c [%2 d] = %g\n", tid , i, c[i]) ;

est execute par tous les threads.

Printf va afficher tid=-1 ,

L’ordre est aleatoire, car dans le

#pragma omp for schedule est dynamic

Avec sort, les instructions lancement par l’ordre des threads

C’est stable

Avec static, la répartition n’est pas stable.

L’avantage de dynamic est de l’ordonnanceur va attribuer dynamiquement et de manière iterative a chaque thread. Meilleur d’utilisation des ressources.

Ex1.3

Oui, le résultat est cohérent

Chaque thread execute 25 fois c[i] = a[i] + b[i]; et d[i] = a[i] + c[i];

Oui, c’est raisonnable, il peut mieux utiliser les ressources.

Non, avec guided le resultat n’est pas coherent

EX1.4

Sans private(val)

La val est toujours 41

Avec private(val)

Il existe toujours une val 38

Le thread private cad, il existe une valeur prive dans le thread. Les autres threads ne peut pas accéder dans ce thread.

C’est impossible

EX1.5

Oui, les résultats sont qu’on attendue

Dans la deux region parallèle, le rprivate est l’autre valeur que 1er region parallèle.