1.- 1718Q2

1-Dibuixar TDG a partir de codi
2.1-Exemple codi recursiu amb tree decomposition
2.2-Task generation control basat en depth
2.3-Task generation control basat en tasques restants per ser executades

2.- 1718Q1

1.1-Dibuixar matriu a partir de codi
1.2-Mirar dependencies en una matriu
1.3-Paral·lelitzar am “paralel” “single” “task” “taskwait”
1.4-Paralelitzar amb task depende
1.5-Modificacio del 1.4
1.6-Data sharing amb atomic

2.1-taskloop
2.2-equivalent amb task

3.1-T1, Tinf, par
3.2-parallel fraction omega
3.3-T4 i S4
3.4.a-Calcul temps en cas recursiu

3.- 1617Q2

1.1-T1, Tinf,Par
1.2-omega i Sinf

2.1-T4 i S4 amb part no paralelitzable

3.1-Calcular overhead a partir de formula reads i writes
3.2-Cronograma

4.-Mecanisme cut-off, limitació de creacio de tasques

4.- 1920Q1

1. Calcular T1 Tinf en un TDG, computar Pmin i mapping de tasques

2.1. corregir codi
2.2. Pregunta teoria sobre load imbalance
2.3. Paralelitzar codi fent servir explicit task
2.4. Paralelitzar utilitzant omp locks

3.1. Modificar codi en un programa recursiu per a tree generation
3.2. Mecanismes de contol en codi reccursiu
3.3. Pregunta de teoria sobre MAX_DEPTH

5.- 1819Q2

1.1. Cambiar codi per reduir overheads
1.2. Reescriure codi per a maximitzar paralelisme usan locks
1.3. Sincronitzacio entre actualitzacions de variables

2. T1 Tinf i exercicis amb el mapping de 3 TDG

3.1. Modificar codi iteratiu per maximitzar load balance sense dependencies
3.2. Same pero en dependencies

4.1. Codi recursiu implementar Tree strategy
4.2. Mecanismes de cut-off en tree

6.- 1819Q1

1.1. T1,Tinf,Par
1.2. Task overhead, creacio de tasques
1.3. Dibuixar cronograma
1.4. Variant Overhead

2.1. Paralelitazcio de codi (crea tasques quan els predecesors acaben)
2.2. Dependencies utilitzant el depend(in: … out : …)

3. Completar codi de varies funcions fent sincronitzacio