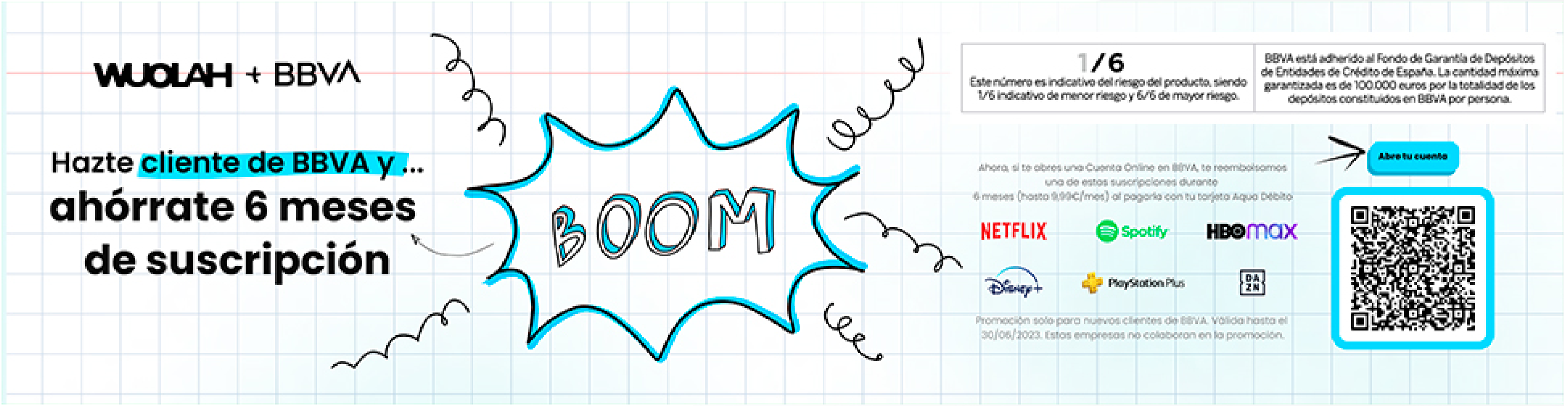
Questions-after-video-lesson-6.pdf

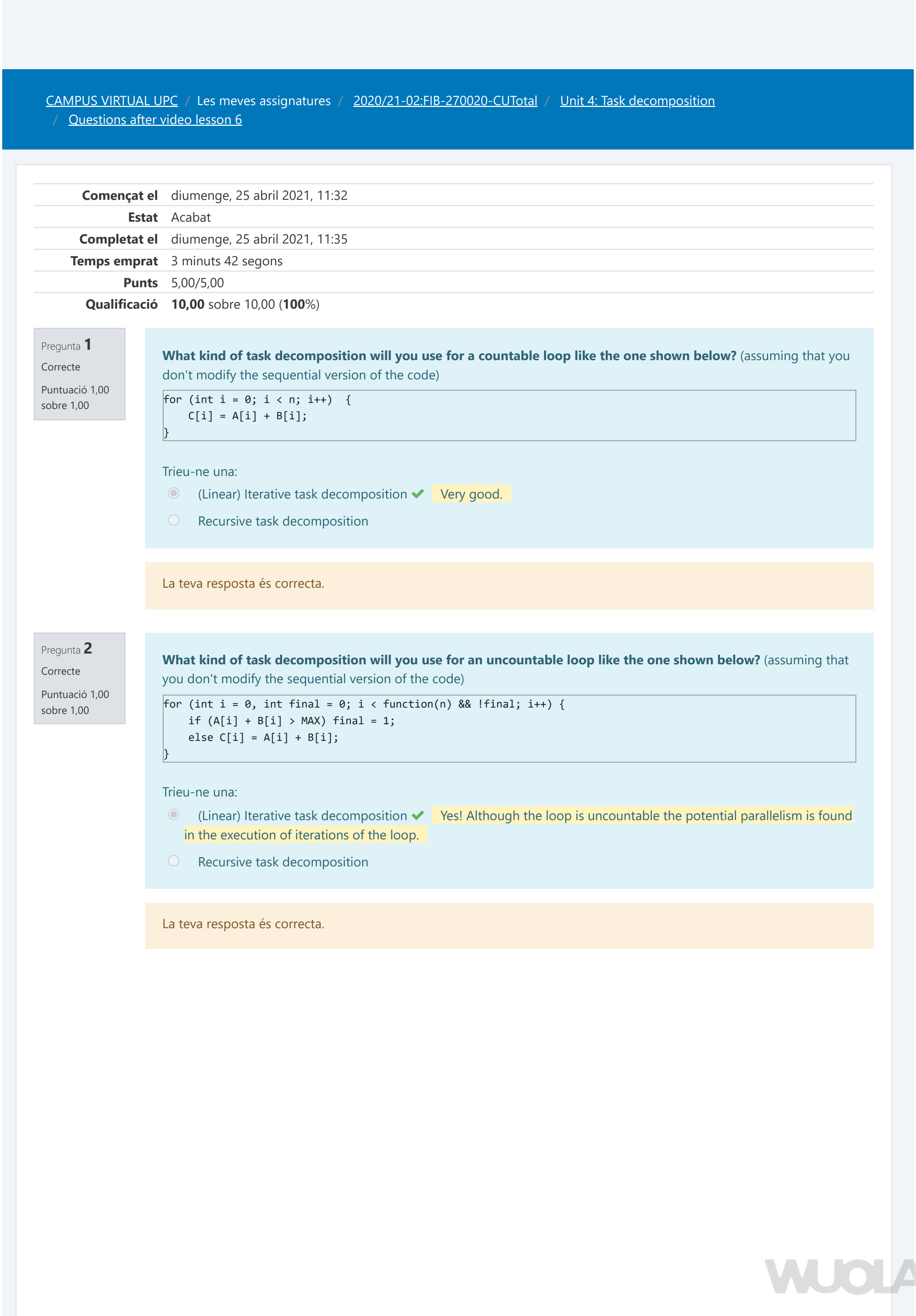
Arnau_FIB

Paralelismo

3º Grado en Ingeniería Informática

Facultad de Informática de Barcelona (FIB)
Universidad Politécnica de Catalunya


WUOLAH + BBVA
Hazte cliente de BBVA y ...
ahórrate 6 meses de suscripción
BOOM
1/6
Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.
BBVA está adherido al Fondo de Garantía de Depósitos de Entidades de Crédito de España. La cantidad máxima garantizada es de 100.000 euros por la totalidad de los depósitos constituidos en BBVA por persona.
Abre tu cuenta
Ahora, si te abres una Cuenta Online en BBVA, te reembolsamos una de estas suscripciones durante 6 meses (hasta 9,99€/mes) al pagarla con tu tarjeta Aqua Débito
NETFLIX
Spotify
HBO max
Disney+
PlayStation Plus
DAZN
Promoción solo para nuevos clientes de BBVA. Válida hasta el 30/06/2023. Estas empresas no colaboran en la promoción.



CAMPUS VIRTUAL UPC / Les meves assignatures / 2020/21-02:FIB-270020-CUTotal / Unit 4: Task decomposition / Questions after video lesson 6

Començat el	diumenge, 25 abril 2021, 11:32
Estat	Acabat
Completat el	diumenge, 25 abril 2021, 11:35
Temps emprat	3 minuts 42 segons
Punts	5,00/5,00
Qualificació	**10,00** sobre 10,00 (**100**%)

Pregunta **1**
Correcte
Puntuació 1,00 sobre 1,00

What kind of task decomposition will you use for a countable loop like the one shown below? (assuming that you don't modify the sequential version of the code)

```
for (int i = 0; i < n; i++)  {
    C[i] = A[i] + B[i];
}
```

Trieu-ne una:

● (Linear) Iterative task decomposition ✓ Very good.

○ Recursive task decomposition

La teva resposta és correcta.

Pregunta **2**
Correcte
Puntuació 1,00 sobre 1,00

What kind of task decomposition will you use for an uncountable loop like the one shown below? (assuming that you don't modify the sequential version of the code)

```
for (int i = 0, int final = 0; i < function(n) && !final; i++) {
    if (A[i] + B[i] > MAX) final = 1;
    else C[i] = A[i] + B[i];
}
```

Trieu-ne una:

● (Linear) Iterative task decomposition ✓ Yes! Although the loop is uncountable the potential parallelism is found in the execution of iterations of the loop.

○ Recursive task decomposition

La teva resposta és correcta.

Pregunta **3**

Correcte

Puntuació 1,00 sobre 1,00

What kind of task decomposition will you use to parallelize the execution of the following function?

```
void
function_increment(int * vector, int n) {
   int n2= n/2;
   if (n==0) return;
   if (n==1) vector[0]++;
   else {
     function_increment(vector,n2);
     function_increment(vector+n2,n-n2);
   }
}
```

Trieu-ne una:

● Recursive task decomposition ✓ Great. Now the question is how? :)

○ (Linear) Iterative task decomposition

La teva resposta és correcta.

Pregunta **4**

Correcte

Puntuació 1,00 sobre 1,00

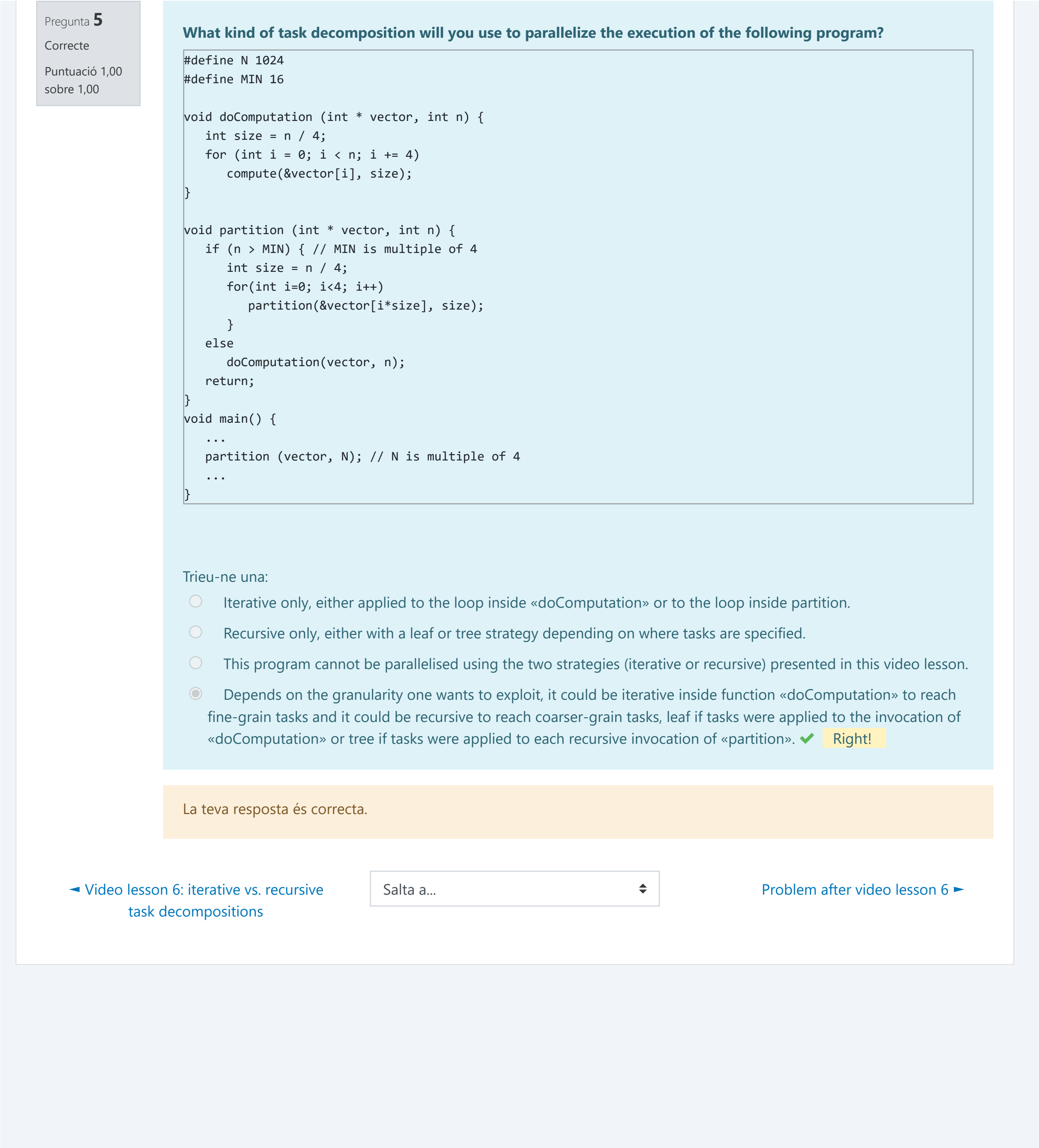
Let's remember the differences between Leaf and Tree Recursive Task Decompositions.

Trieu-ne una o més:

☑ Leaf Recursive task decompositions allow the exploitation of the parallelism among all the tasks that are created for the leaves in a tree recursive traversal. ✓ Right!

☑ Tree Recursive task decompositions parallelize the traversal of the tree, usually reducing the overall parallel execution time. ✓ Right!

La teva resposta és correcta.



Pregunta **5**

Correcte

Puntuació 1,00 sobre 1,00

What kind of task decomposition will you use to parallelize the execution of the following program?

```
#define N 1024
#define MIN 16

void doComputation (int * vector, int n) {
   int size = n / 4;
   for (int i = 0; i < n; i += 4)
      compute(&vector[i], size);
}

void partition (int * vector, int n) {
   if (n > MIN) { // MIN is multiple of 4
      int size = n / 4;
      for(int i=0; i<4; i++)
         partition(&vector[i*size], size);
      }
   else
      doComputation(vector, n);
   return;
}
void main() {
   ...
   partition (vector, N); // N is multiple of 4
   ...
}
```

Trieu-ne una:

○ Iterative only, either applied to the loop inside «doComputation» or to the loop inside partition.

○ Recursive only, either with a leaf or tree strategy depending on where tasks are specified.

○ This program cannot be parallelised using the two strategies (iterative or recursive) presented in this video lesson.

● Depends on the granularity one wants to exploit, it could be iterative inside function «doComputation» to reach fine-grain tasks and it could be recursive to reach coarser-grain tasks, leaf if tasks were applied to the invocation of «doComputation» or tree if tasks were applied to each recursive invocation of «partition». ✓ Right!

La teva resposta és correcta.

◄ Video lesson 6: iterative vs. recursive task decompositions

Problem after video lesson 6 ►