## $ext{PAR} - ext{In-Term Exam} - ext{Course } 2023/24 ext{-} ext{Q2} \ ext{April } 4^{th}, \ 2024$

**Problem 1** (4.0 points) Given the following code:

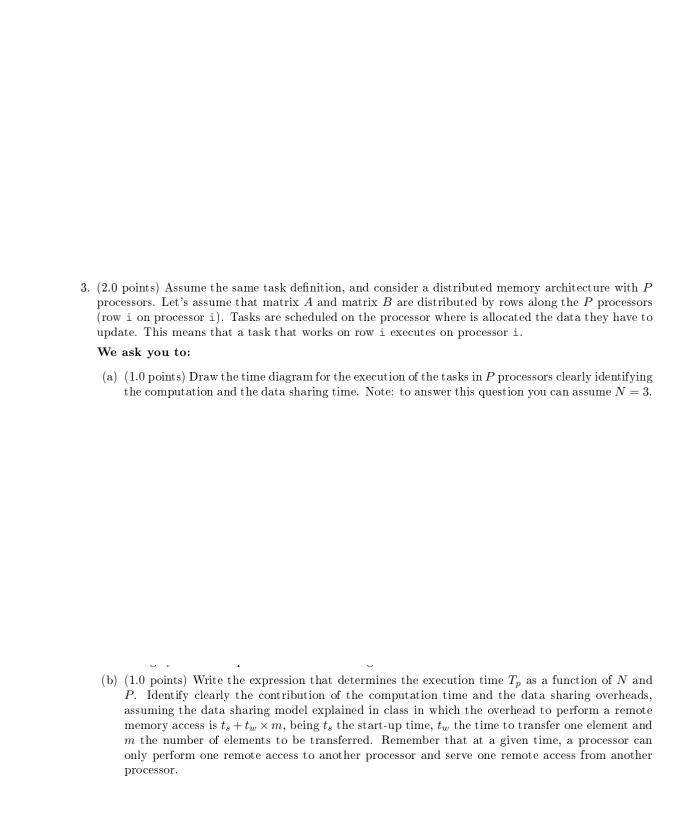
```
\#define N ... // value determined at each section
float A[N][N], B[N][N];
// initialization
for (int i=0; i<N; i++) {
   tareador_start_task("init");
   for (k=0; k<N; k++) {
       A[i][k] = init(); // 10 time units
  tareador_end_task("init");
// calculation
for (int i=1; i<N; i++) {
   tareador_start_task("odd"); // Process only odd columns
   for (k=1; k<N; k+=2) {
       B[i][k] = A[i-1][k] + A[i][k] + foo(); // 20 time units
   tareador_end_task("odd");
   tareador_start_task("even"); // Process only even columns
   for (k=2; k<N; k+=2) {
       B[i][k] = A[i-1][k] + A[i][k] + B[i][k-1] + goo(); // 40 time units
   tareador_end_task("even");
```

Assuming that functions foo, goo and init do not modify any element from matrix A and matrix B, we ask you to:

1. (1.0 points) Draw the Task Dependence Graph (TDG) based on the Tareador task definitions in the instrumented code above. Each task should be clearly labeled with the value of i and its cost in time units. Assume that N=3.

2. (1.0 points) Compute the values for  $T_1$ ,  $T_{\infty}$  and  $P_{min}$ . Draw the temporal diagram for the execution of the TDG if executed using  $P_{min}$  processors.

1



**Problem 2** (3.0 points) Consider the following sequential code:

```
object_type *mat[256][256];
int histo[5] = {0, 0, 0, 0, 0};
int object_val(object_type *p); // return value of object *p in the range 0..4

int main() {
    for (int i=0; i<256; i++)
        for (int j=0; j<256; j++)
            histo[object_val(mat[i][j])]++;
}</pre>
```

that computes the histogram of the values in the range [0..4] related to the objects pointed by a square matrix of  $256 \times 256$  elements.

We ask you to: Write an iterative task decomposition strategy using OpenMP that efficiently exploits the parallellism reducing task creation and synchronization overheads. You can propose a solution based on implicit or explicit tasks.

## **Problem 3** (3.0 points) Given the following sequential code:

## We ask you to:

- 1. (1.5 points) Create a parallel version in OpenMP using a recursive task decomposition for the fibbonacci function. Select the most appropriate strategy (tree or leaf) that will maximize the processor utilisation assuming a system with a high number of processors and without considering task creation and synchronization overheads.
- 2. (1.0 points) Modify the previous code to implement a task generation control mechanism based on the depth level. Use MAX\_DEPTH as the maximum depth level to decide if tasks must be created or not.

3.	. (0.5 points) Assume your first parallel version. Which type of synchronization is required to guar that your tasks have finished before the printf statement in the main program if you can not as any implicit or explicit thread barrier between the fibbonacci and printf calls?.	antee sume