Solutions should be typed or very very neatly printed. Diagrams may be hand-drawn neatly. As much as appropriate, use the design notation from Andrews’s text and the course notes.

Q0 [10]
(a) Design an algorithm to sort an array \(a\) of length \(N\) using a merge sort. Assume you have \(N/2\) processes. Your algorithm should take \(\Theta(N)\) time.
(b) Design merge sort as a bag of tasks algorithm.

Q1 [30] Rooms
There are two rooms and any number of threads. The rules are that

- if any thread is in room 0, no thread may be in room 1; and
- if any thread is in room 1, no thread may be in room 0.

Each thread strictly alternates entering and leaving rooms:

```c
while (true) {
    do something private
    i := either 0 or 1
    enter(i);
    do something in the room
    leave(i);
}
```

(a) Create a high-level design for enter and leave using await statements.
(b) Using semaphores, implement the enter and leave routines.
(c) Design a monitor called RoomMonitor that exports enter and leave. (We change the calls above to RoomMonitor.enter() and RoomMonitor.leave().) The monitor must use the Signal and Wait signalling discipline. All condition variables must be documented with their corresponding condition.

Q2 [25]
(a) Design a parallel prefix algorithm for the following problem. Given a boolean array \(b\) of length \(N\), compute two other arrays, each of length \(N\):

- \(ntb(i)\) is the number of true items that occur strictly before item \(i\) of \(b\).
- \(nfa(i)\) is the number of false items that occur strictly after item \(i\) of \(b\).
For example (primes indicate final values):

\[
\begin{array}{c|c|c|c|c|c|c|c}
\text{b} & \text{true} & \text{false} & \text{true} & \text{false} & \text{false} & \text{true} & \text{true} \\
\hline
\text{ntb'} & 0 & 1 & 1 & 2 & 2 & 2 & 3 \\
\text{nfa'} & 3 & 2 & 2 & 1 & 0 & 0 & 0
\end{array}
\]

(b) Partitioning is a useful activity, for example it is used in Hoare’s quick-sort algorithm and his algorithm for finding the k’th largest item. Given a double array \( a \) of length \( N \) and a double \( p \), partitioning permutes \( a \) so that all items less or equal to \( p \) are to the left of all items greater than \( p \). For example, with \( p = 2.5 \),

\[
\begin{array}{c|c|c|c|c|c|c|c}
\text{a} & 2.5 & 3.0 & 1.5 & 4.5 & 5.0 & 2.5 & 0.5 \\
\hline
\text{a'} & 2.5 & 1.5 & 2.5 & 0.5 & 3.0 & 4.5 & 5.0
\end{array}
\]

(In the example, I maintained the order of values as much as I could. This is not required.) Design a fast parallel algorithm to partition an array. Use \( N \) processors to achieve \( O(\lg N) \) time.

(c) We couldn’t afford \( N \) processors. We only have \( P \) processors. Outline how your solution to part (b) changes. In terms of \( P \) and \( N \), what is the time complexity of your new solution?

**Q3 [10] (Bonus) Search**

We wish to search a genome for patterns specified by regular expressions. For this question, you will merely design an algorithm that will find all occurrences of the pattern \( AC^*T \), i.e. an \( A \) followed by any number (including 0) of \( C \)s, followed by a \( T \).

We will use a massively parallel computer with \( N \) processors to search a string \( s \) of \( N \) letters in time \( O(\lg N) \).

Your algorithm should compute into an array item \( b(i) \), the boolean indicating whether or not the pattern starts at position \( i \) of \( s \).

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
\text{s} & G & A & T & A & C & A & C & C & T & A & C \\
\hline
\text{b'} & \text{false} & \text{true} & \text{false} & \text{false} & \text{false} & \text{true} & \text{false} & \text{false} & \text{false} & \text{false} & \text{false}
\end{array}
\]

**Hint.** See the solution to last year’s assignment 1.