Problem 1. In our description of the Josephus problem we have participants standing in a circle and one sword. One participant has a sword. In one move, the participant with the sword kills the next participant in the counterclockwise direction, the cycle closes and the sword is passed to the next participant in the counterclockwise direction. Our goal is to determine who will survive if \( n \) participants start and we number/name them from 0 to \( n - 1 \). Assume that participants number 0 is the one who has the sword at the beginning.

(a) Show that if \( n \) is a power of 2, participant 0 is the survivor. Use mathematical induction.

(b) Show that if \( n = p + r \) where \( p \) is a power of 2 and \( r < p \) then the survivor is participant \( 2r \). Hint: who gets the sword when the number of participants drops to \( p \)?

Problem 2. Recursive mergesort can be more efficient than iterative mergesort but copying before sorting is a major drain of running time. We use notation \((A, i, j)\) for array sector \( A[i], A[i+1], \ldots, A[j-1] \).

(a) Write code of a function that merges sorted sectors \((A, a, b)\) and \((A, c, d)\) onto \((A, c, d + b - a)\), assuming \( a \leq b \leq c \leq d \). It should move each array entry from these sectors exactly once.

(b) Write recursive code that sorts fragment \((A, a, b)\) so that the result is in fragment \((A, b - 1, 2b - a - 1)\). It should move array entries only using the calls to the merging function.

(c) Explain how large array \( A \) should be to sort \((A, 0, n)\) using this method.

Problem 3. You have as the input a connected undirected graph.

(a) Can you always give a single direction to each edge so the graph remains strongly connected? Show a counterexample.

(b) Sketch an algorithm to identify edges that make the graph not-strongly connected if we give them a single direction.

Problem 4. Order the following functions so when \( f_i \) is followed by \( f_j \) then \( f_i = O(f_j) \). Indicate the cases when \( f_i = \Theta(f_j) \).

\[
\begin{align*}
 f_1(n) &= (\ln n)^n \\
 f_2(n) &= (\ln n)! \\
 f_3(n) &= (n)^{\ln n} \\
 f_4(n) &= 5^n \\
 f_5(n) &= n \\
 f_6(n) &= 2^{\ln n}
\end{align*}
\]

Note: To receive partial credit for a not entirely correct answer, you must show your work neatly, and your work should not simply consist of evaluating the functions at a large value of \( n \) in the hope that this value is large enough to reveal the correct answer.