1. If a red-black tree of \( n \) nodes is created by using RB-INSERT, is it true that the resulting tree always contains red nodes, if \( n > 1 \)?

\[
\text{RB-INSERT}(T, z) \quad \text{RB-INSERT-FIXUP}(T, z)
\]

\[
y = T.nil \quad \text{while } z.p.color == \text{RED} \text{ do}
\]

\[x = T.root \quad \text{if } z.p == z.p.p.left \text{ then}
\]

\[ \text{while } x \neq T.nil \text{ do} \quad \text{if } y.color == \text{RED} \text{ then}
\]

\[y = x \quad z.p.color = \text{BLACK}
\]

\[\text{if } z.key < x.key \text{ then} \quad y.color = \text{BLACK}
\]

\[x = x.left \quad z.p.p.color = \text{RED}
\]

\[\text{else} \quad y = z.p \quad z = z.p.p
\]

\[x = x.right \quad \text{else if } z == z.p.right \text{ then}
\]

\[z.p.color = \text{BLACK}
\]

\[\text{if } y.key < x.key \text{ then} \quad z.p.p.color = \text{RED}
\]

\[x.left = z \quad \text{RIGHT-ROTATE}(T, z.p.p)
\]

\[\text{else} \quad y.left = z \quad \text{same as the then clause with}
\]

\[y.right = z; \quad \text{"right" and "left"}
\]

\[z.left = T.nil \quad \text{exchanged}
\]

\[z.right = T.nil \quad \text{T.root.color} = \text{BLACK}
\]

\[z.color = \text{RED} \quad \text{returns}
\]

\[\text{RB-INSERT-FIXUP}(T, z)
\]

2. If \( n \) elements, numbered \( 1 \)–\( n \), form a circle, and starting counting from Element 1, one proceeds around and around removing every \( m^{\text{th}} \) \( (m \leq n) \) element until all the elements have been removed, the removal order gives us the \((n, m)\)-Josephus permutation of the integers 1, 2, ..., \( n \). Describe an algorithm that, given integers \( n \) and \( m \), outputs the \((n, m)\)-Josephus permutation in \( O(n \log n) \) time. (Hint: use a suitable data structure discussed in the lectures.)

3. Let us modify the rod-cutting problem so that cutting is not free anymore: one has to pay a fixed cost \( c \) for each cut, which affects the revenues of cutting solutions. The value for a solution is the sum of the piece prices reduced by the costs of making the cuts to obtain the pieces. Modify MEMO-CUT-ROD and MEMO-CUT-ROD-AUX in order to take the cutting costs into account.

\[
\text{MEMO-CUT-ROD}(p, n)
\]

\[
\text{MEMO-CUT-ROD-AUX}(p, n, r)
\]

\[
\text{let } r[0..n] \text{ be a new array}
\]

\[
\text{if } r[n] \geq 0 \quad \text{if } n = 0
\]

\[
\text{return } r[n] \quad q \leftarrow 0
\]

\[
\text{for } i \leftarrow 1 \text{ to } n \quad \text{else } q \leftarrow -\infty
\]

\[
r[i] \leftarrow -\infty
\]

\[
\text{return } \text{MEMO-CUT-ROD-AUX}(p, n, i, r)
\]

\[
r[n] \leftarrow q
\]

4. Continue modifying the algorithm obtained as the result for Question 3 and make it (in addition to the optimal value) also return the actual cutting solution.