Exercise 1: Fibonacci Heaps (5 points)
Execute the following operations on an initially empty Fibonacci heap:

\text{insert}(15), \text{insert}(27), \text{insert}(6), \text{insert}(34), \text{insert}(42), \\
\text{insert}(35), \text{insert}(3), \text{insert}(41), \text{insert}(22), \text{insert}(12), \\
\text{deletemin}(), \text{decreasekey}(27, 2), \text{decreasekey}(34, 17), \text{deletemin}() \\

For all intermediate stages, illustrate the structure of the Fibonacci heap, fill in the key values and possible marks of the nodes, and tag the current minimum. A new element shall always be inserted to the right of the current minimum. The consolidation during the operation \text{deletemin} shall start with the element to the right of the deleted minimum.

Exercise 2: Fibonacci Heaps (5 points)
Show that the following claim is not true:

The maximum height of a tree within a Fibonacci heap with \( n \) nodes is \( O(\log n) \).

Proceed as follows: For an arbitrary \( n > 0 \) give a sequence of operations that creates a Fibonacci heap finally consisting of one tree that is a linear chain of \( n \) nodes.

Exercise 3: Linked-list representation for disjoint sets. (5 points)

a) Write pseudocode for the procedures \text{make-set}, \text{find-set}, and \text{union} using the linked-list representation of disjoint sets. Apply the weighted-union heuristic in the \text{union} procedure. Assume that each element \( x \) has four attributes:

\begin{itemize}
  \item \text{x.next} : pointer to the next element of the list; \text{nil} if \( x \) is the last element
  \item \text{x.rep} : pointer to the set representative, that is, to the first element of the list
  \item \text{x.last} : if \( x \) is the first element of a list, then this field points to the last element
  \item \text{x.size} : if \( x \) is the first element, then this field contains the size of the list
\end{itemize}

If \( x \) is not the first element of a list, then its \text{last} and \text{size} fields are not used by any of the procedures, and the information in these fields may be incorrect.

b) Modify the \text{union} procedure from a) such that it is no longer necessary to keep the \text{last} pointer. Your modification should not change the asymptotic running time.

\text{Hint}: Rather than appending one list to another, splice them together.
Exercise 4: Disjoint-set forests (5 points)
Write pseudocode for a nonrecursive version of the find-set procedure for disjoint-set forests that uses the path-compression heuristic. The procedure shall traverse the find path (the path from the element on which the procedure is called toward the root) at most twice.

*Hint:* You may use a stack $S$ with operations push, pop, and isEmpty.

The following exercise gives 5 extra points:

Exercise 5: Connected-Components (5 points)
Let $G = (V, E)$ be an undirected graph with $k$ connected components.

a) Show that after all edges are processed by algorithm Connected-Components, two vertices are in the same connected component if and only if they are in the same set.

b) During the execution of Connected-Components on $G$, how many times is the procedure find-set called? How many times is union called? Express your answers in terms of $|V|$, $|E|$, and $k$ and prove them.