**Problem 1**

A *d-ary heap* is like a binary heap, but instead of 2 children, nodes have *d* children.

(a) How would you represent a *d*-ary heap with *n* elements in an array? What are the expressions for determining the parent of a given element, \( \text{Parent}(i) \), and a *j*-th child of a given element, \( \text{Child}(i, j) \), where \( 1 \leq j \leq d \)?

The following expressions determine the parent and *j*-th child of element *i* (where \( 1 \leq j \leq d \)):

\[
\begin{align*}
\text{Parent}(i) &= \left\lfloor \frac{i + d - 2}{d} \right\rfloor, \\
\text{Child}(i, j) &= (i - 1) \cdot d + j + 1.
\end{align*}
\]

(b) Write an efficient implementation of \texttt{Heapify} and \texttt{Heap-Insert} for a *d*-ary heap.

The \texttt{Heapify} algorithm is somewhat different from the binary-heap version, whereas \texttt{Heap-Insert} is identical to the corresponding algorithm for binary heaps. The running time of \texttt{Heapify} is \( O(d \cdot \log_d n) \), and the running time of \texttt{Heap-Insert} is \( O(\log_d n) \).

\[
\begin{align*}
\text{Heapify}(A, i, n, d) \\
&\quad \text{largest} \leftarrow i \\
&\quad \text{for } l \leftarrow \text{Child}(i, 1) \text{ to } \text{Child}(i, d) \triangleright \text{ loop through all children of } i \\
&\quad \quad \text{do if } l \leq n \text{ and } A[l] > A[\text{largest}] \\
&\quad \quad \quad \text{then } \text{largest} \leftarrow l \\
&\quad \quad \text{if } \text{largest} \neq i \\
&\quad \quad \quad \text{then exchange } A[i] \leftrightarrow A[\text{largest}] \\
&\quad \quad \quad \text{Heapify}(A, \text{largest})
\end{align*}
\]

\[
\begin{align*}
\text{Heap-Insert}(A, key) \\
&\quad \text{heap-size}[A] \leftarrow \text{heap-size}[A] + 1 \\
&\quad i \leftarrow \text{heap-size}[A] \\
&\quad \text{while } i > 1 \text{ and } A[\text{Parent}(i)] < key \\
&\quad \quad \text{do } A[i] \leftarrow A[\text{Parent}(i)] \\
&\quad \quad \quad i \leftarrow \text{Parent}(i) \\
&\quad A[i] \leftarrow key
\end{align*}
\]

**Problem 2**

What is the height of a *d*-ary heap of *n* elements in terms of *n* and *d*?

The height *h* of a heap is *approximately* equal to \( \log_d n \). The exact height is

\[
h = \lceil \log_d(n \cdot d - n + 1) - 1 \rceil.
\]