Exercise 1: (Binary) Heaps and Heapsort \hspace{1cm} (12 Bonus Points)

(a) Implement a binary heap using the array implementation from the lecture. The heap should support the following functions: `create`, `insert` (eines key-value pairs), `get_min` and `delete_min`. You may use the template `heap.py`. \hspace{1cm} (5 Points)

\textit{Hint:} To implement \texttt{delete_min} efficiently one overwrites the root with the last element of the heap and then deletes the last element. Afterwards one has to repair the min-heap property.

(b) Implement the heapsort algorithm by using your implementation from the previous task.\footnote{If you did not solve the previous task, you may use \texttt{heapq}. In \texttt{heapq}, \texttt{heappush} equals the `insert` and \texttt{heappop} the `delete-min` operation from the lecture. \texttt{heappush} and \texttt{heappop} can be applied on Python-lists (for more detail see \texttt{here}).} Begründen Sie warum Heapsort eine Laufzeit von $O(n \log n)$ hat.

Explain why there can be no heap implementation where `insert`, `get_min` and `delete_min` have all constant runtime. \hspace{1cm} (3 Points)

(c) In this task we consider ternary heaps. The are similar to binary heaps with the difference that each parent node may have 3 children. Again we assume that the tree is completely filled (from top to down and left to tight).

Give the minimal and maximal number of nodes of a ternary heap of depth $d$. \hspace{1cm} (1 Point)

Assume we use an array implementation as for binary heaps, starting with index 1 (not 0). Let $i$ be the index of a node $v$ that is neither the root nor a leaf. What are the indices of $v$'s parent and its three children? \hspace{1cm} (3 Points)
Exercise 2: Hashing

(a) Let \( h(s, j) := h_1(s) - 2j \mod m \) and \( h_1(x) := x + 2 \mod m \). Insert the keys 51, 13, 21, 30, 23, 72 (in the given order) into a hash table of size \( m = 7 \) by using the hash function \( h \) and linear probing for collision resolution. (The following table should show the final state after inserting all keys.)

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]

(1 Point)

(b) Assume we would like to insert the sequence of numbers from part a) in a table of size \( m = 7 \) by using quadratic probing. Which of the following hash functions would be the better choice? Explain your answer.

- \( h_1(x, i) := x + 6i + 2i^2 \mod m \)
- \( h_2(x, i) := x + i + 4i^2 \mod m \)

Insert the keys by using the better hash function into the following table.

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]

(2 Points)

(c) Let \( h(s, j) := h_1(s) + j \cdot h_2(s) \mod m \) with \( h_1(x) = x \mod m \) and \( h_2(x) = 1 + (x \mod (m - 1)) \). Insert the keys 28, 59, 47, 13, 39, 69, 12 in a hash table of size \( m = 11 \) by using double-hashing for collision resolution.

\[
\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\end{array}
\]

(2 Points)

(d) Given the hash functions \( h_1(x) := x + 2 \mod m \) and \( h_2(x) := 3x \mod m \) with \( m = 7 \), find three pairwise distinct keys \( u, v, w \in \mathbb{N} \) such that \( h_1(u) = h_1(v) = h_1(w) \neq h_2(u) = h_2(v) = h_2(w) \). Insert \( u \) and \( v \) into the following table by using Cuckoo Hashing.

\[
\begin{array}{ccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{array}
\]

If one also inserts \( w \), one obtains a cycle. To avoid this, one makes a rehash by increasing the table’s size to \( m' = 11 \) and use two new hash functions \( h'_1 \) und \( h'_2 \). Give the functions \( h'_1 \) und \( h'_2 \) such that \( u, v \) and \( w \) can be inserted into the new table without obtaining a cycle (\( h'_1 \) and \( h'_2 \) should be different hash function of the form \( (ax \mod m') \) with \( a \neq 0 \) sein.).

(3 Points)