Problem 1. Fill in the blanks regarding heaps:

For a node at index \( p \), \( p \)'s children are at indices \( \text{function of } p \) and \( \text{function of } p \).

For a node at index \( c \), \( c \)'s parent is at index \( \text{function of } c \).

Index 10 of the array corresponds to a node at depth \( \text{number} \).

The leftmost index corresponding to a node at depth 2 is \( \text{number} \).

Problem 2. Fill in the blanks regarding a heap implemented via a ternary approach (each node has up to three children):

For a node at index \( p \), \( p \)'s children are at indices \( \text{function of } p \), \( \text{function of } p \), and \( \text{function of } p \).

For a node at index \( c \), \( c \)'s parent is at index \( \text{function of } c \).

Index 10 of the array corresponds to a node at depth \( \text{number} \).

The leftmost index corresponding to a node at depth 2 is \( \text{number} \).
Problem 3. Fill in the array corresponding to the min heap in Figure 1:

```
0  1  2  3  4  5  6  7  8
```

Do the same, but for the array after calling `push(1)` on the heap:

```
0  1  2  3  4  5  6  7  8  9
```

Do the same, but for the array after calling `pop()` on the heap:

```
0  1  2  3  4  5  6  7
```

Do the same, but for the array after calling `pop()` twice on the heap:

```
0  1  2  3  4  5  6
```

Figure 1: The heap for Problem 3.
Problem 4. Fill in the array corresponding to the max heap in Figure 2:

Do the same, but for the array after calling \texttt{push(1)} on the heap:

Do the same, but for the array after calling \texttt{pop()} on the heap:

Do the same, but for the array after calling \texttt{pop()} \textbf{twice} on the heap:

Figure 2: The heap for Problem 4.