Implement a Min Heap

This program showcases the implementation of a Min-Heap, a vital data structure commonly used in various algorithms and applications. A Min-Heap is a specialized binary tree where each parent node has a value smaller than its children. This structure ensures that the smallest element is always at the root, making it efficient for tasks that require quick access to the minimum value.

Example: [55,33,44,22,11,40,66]

```
        11
       /  \
     22    40
    / \   / \
  55  33 44  66
```

1. **MinHeap Class:**
    - The **MinHeap** class encapsulates the Min-Heap data structure.
    - Key Attributes:
        - **arr**: A dynamic array for storing heap elements.
        - **size**: Current number of elements in the heap.
        - **capacity**: Initial capacity of the dynamic array.
2. **Constructor:**
    - The class constructor initializes the **MinHeap** object. It allocates memory for the dynamic array, setting the initial size and capacity.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
3. **Insertion Function:**
    - **insert(int value)**: Adds a new element to the Min-Heap.
        - If the array is full, it dynamically reallocates memory to double the capacity.
        - After insertion, it ensures that the Min-Heap property is maintained by comparing the element with its parent and performing swaps if necessary.

- **Time Complexity: O(log n) in the worst case, where 'n' is the number of elements in the heap.**
- **Space Complexity: O(1) (amortized)**

4. **Deletion Function:**
    - **deleteHeapElement()**: Removes the minimum element from the Min-Heap.
        - The function replaces the root with the last element and then "heapifies" the tree to ensure the Min-Heap property is preserved. It compares the root with its children and swaps as needed until the smallest element is at the top.
    - **Time Complexity: O(log n) in the worst case, where 'n' is the number of elements in the heap.**
    - **Space Complexity: O(1)**
5. **Print Function:**
    - **printHeap()**: Displays the elements in the Min-Heap.
        - It checks if the heap is empty and prints the elements if not.
    - **Time Complexity: O(n), where 'n' is the number of elements in the heap.**
    - **Space Complexity: O(1)**
6. **Minimum Element Retrieval:**
    - **getMin()**: Returns the minimum element of the Min-Heap.
        - It checks if the heap is empty and returns -1 if so.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**
7. **Destructor:**
    - The class destructor deallocates memory for the dynamic array when the **MinHeap** object goes out of scope.
    - **Time Complexity: O(1)**
    - **Space Complexity: O(1)**