Implement a Max Heap

This program demonstrates the implementation of a Max-Heap, a fundamental data structure used in various algorithms and applications. A Max-Heap is a specialized binary tree where each parent node has a value greater than its children. This ensures that the largest element is always at the root, making it efficient for tasks like finding the maximum element quickly.

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

66

/ \

33 55

/ \ / \

22 11 40 44

1. **MaxHeap Class:**
   * The **MaxHeap** class encapsulates the Max-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 **MaxHeap** 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 Max-Heap.
     + If the array is full, it dynamically reallocates memory to double the capacity.
     + After insertion, it ensures that the Max-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 maximum element from the Max-Heap.
     + The function replaces the root with the last element and then "heapifies" the tree to ensure the Max-Heap property is preserved. It compares the root with its children and swaps as needed until the largest 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 Max-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. **Maximum Element Retrieval:**
   * **getMax()**: Returns the maximum element of the Max-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 **MaxHeap** object goes out of scope.
   * **Time Complexity:** **O(1)**
   * **Space Complexity: O(1)**