Constructor Overloading and Initialization:

The ArrayHeap class has two constructors, one default and one parameterized. The default constructor initializes the heap with a capacity of 10. It would be beneficial to initialize the heap array dynamically based on the capacity to avoid fixed-size arrays and make the heap more flexible.

Method Implementations:

addElement(int elem): This method adds an element to the heap and ensures the heap property is maintained by bubbling up. It's crucial to note that the method assumes the dataValue is the string representation of the priorityValue. This might not always be the desired behavior in a priority queue where data could be different from priority values.

removeMin(int i): The method name is misleading as it suggests removing the minimum element, but it actually removes an element at a specific index i. This could be renamed to removeAt(int index) for clarity. Additionally, the method should handle the case where i is out of bounds.

returnMin(int i): Similar to removeMin, the method name is misleading. It returns the element at index i rather than the minimum element. Renaming this to getElementAt(int index) would be clearer.

toString(): This method is intended to return a string representation of the heap, but it currently prints directly to std::cout and returns an integer due to a misplaced return 0;. This should be corrected to build and return a string instead.

Error Handling:

The code lacks robust error handling, especially in user inputs and file operations. For example, std::stoi(input) is used without checking if input is a valid integer string.

Code Organization and Readability:

The main function is overly long and handles multiple responsibilities, from user input to action processing. Breaking down this function into smaller, more focused functions would improve readability and maintainability.

Consistent naming conventions and more descriptive variable names would enhance readability. For instance, variable names like x and i in loops could be more descriptive.

Memory Management:

The heap array is statically sized, which limits the flexibility of the ArrayHeap class. Consider using std::vector for dynamic array management, which would automatically handle resizing and improve memory usage.

User Interaction:

The loop for action commands (I, R, S, A, Q) could be encapsulated into a separate function to clean up the main function and isolate functionality.

By addressing these points, the code can be made more robust, flexible, and easier to understand and maintain.Test Default Constructor:

Objective: Verify that the default constructor initializes the heap with a capacity of 10 and size 0.

Input: None.

Expected Output: Capacity = 10, Size = 0.

Test Parameterized Constructor:

Objective: Check if the parameterized constructor correctly sets a custom capacity.

Input: Capacity = 20.

Expected Output: Capacity = 20, Size = 0.

Test Add Element:

Objective: Ensure that adding an element updates the size and places the element correctly.

Input: Add element 5.

Expected Output: Heap contains [5], Size = 1.

Test Heap Property on Insert:

Objective: Confirm that the heap property is maintained after multiple inserts.

Input: Add elements 10, 20, 5, 1.

Expected Output: Heap contains [1, 5, 20, 10], Size = 4.

Test Remove Min:

Objective: Verify that the minimum element is removed and heap property is maintained.

Input: From previous state, remove min.

Expected Output: Heap contains [5, 10, 20], Size = 3.

Test Return Min:

Objective: Check if the minimum element is returned without removal.

Input: From current heap state.

Expected Output: Returns 5, Heap remains unchanged.

Test Full Heap:

Objective: Ensure that no more elements can be added once the heap is full.

Input: Fill the heap to capacity and try adding one more element.

Expected Output: "Heap is full!" message, no change in heap content or size.

Test Empty Heap Removal:

Objective: Attempt to remove an element from an empty heap.

Input: Remove from an empty heap.

Expected Output: "Heap is empty!" message, no change in heap.

Test Random Inserts and Remove Operations:

Objective: Perform a series of random insertions and removals to stress test the heap operations.

Input: Random sequence of add and remove operations.

Expected Output: Heap maintains correct size and order properties after each operation.

Test toString Method:

Objective: Check if the toString method outputs the correct string representation of the heap.

Input: Current state of the heap.

Expected Output: Correct string representation of each element in the heap.