# PROGRAM – 9

**Title: BINOMIAL HEAP**  
  
**Objective:** Write a program for binomial heap.  
  
**Algorithm:**  
**1. Define the Min-Heap Structure**  
 • The MinHeap structure consists of:  
 - An array arr to store heap elements.  
 - An integer size to represent the current number of elements in the heap.  
 - An integer capacity to represent the maximum number of elements the heap can hold.  
  
**2. Create a Min-Heap**  
Function: createMinHeap(capacity)  
 1. Allocate memory for a MinHeap structure.  
 2. Initialize:  
 - arr with memory to hold up to capacity elements.  
 - size to 0 (empty heap).  
 - capacity to the given value.  
 3. Return the newly created heap.  
  
**3. Helper Functions**  
Function: parent(i)  
 • Returns the index of the parent of the node at index i.  
  
Function: leftChild(i)  
 • Returns the index of the left child of the node at index i.  
  
Function: rightChild(i)  
 • Returns the index of the right child of the node at index i.  
  
Function: swap(a, b)  
 • Swaps the values of a and b.  
  
**4. Heapify Up**  
Function: heapifyUp(heap, index)  
 1. While the index is greater than 0 and the parent node is greater than the current node:  
 - Swap the current node with its parent.  
 - Update the index to the parent’s index.  
  
**5. Heapify Down**  
Function: heapifyDown(heap, index)  
 1. Determine the left and right child indices of the current node.  
 2. Initialize the smallest element’s index as the current node’s index.  
 3. Compare the left child with the current node:  
 • If the left child exists and is smaller, update smallest to left child’s index.  
 4. Compare the right child with the smallest node:  
 • If the right child exists and is smaller, update smallest to the right child’s index.  
 5. If smallest is not the current node:  
 • Swap the current node with the smallest child.  
 • Recursively call heapifyDown for the updated index.  
  
**6. Insert a Value**  
Function: insert(heap, value)  
 1. Check if the heap is full:  
 • If size == capacity, print an error and return.  
 2. Insert the value at the end of the heap array.  
 3. Increment the heap’s size.  
 4. Call heapifyUp(heap, index) for the last inserted element (index = size - 1).  
  
**7. Extract the Minimum**  
Function: extractMin(heap)  
 1. If the heap is empty (size == 0), print an error and return -1.  
 2. Store the minimum value (min = arr[0]).  
 3. Replace the root element with the last element in the heap.  
 4. Decrement the size.  
 5. Call heapifyDown(heap, index = 0).  
 6. Return the extracted minimum value (min).  
  
**8. Print the Heap**  
Function: printHeap(heap)  
 1. Iterate through the heap array (arr[]) up to the size.  
 2. Print each element.  
  
**9. Main Function**  
Steps:  
 1. Create a Min-Heap with a specified capacity.  
 2. Insert elements into the heap using the insert() function.  
 3. Print the heap after all insertions.  
 4. Extract the minimum value from the heap using extractMin().  
 5. Print the extracted minimum value and the heap after the extraction.