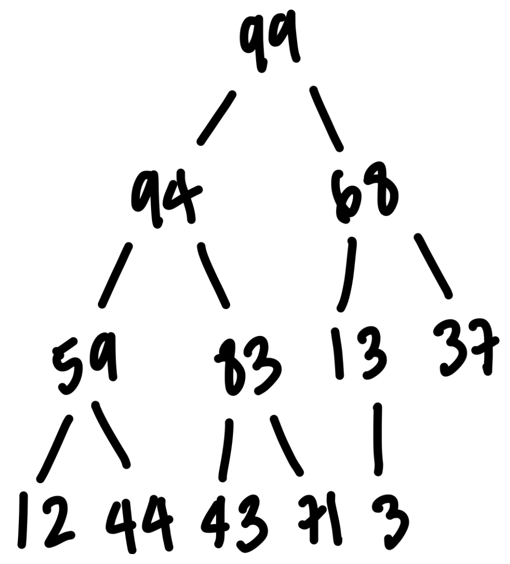
PA 7 Part 1: Heap Worksheet

DSC 30 Fall 2023

Overview: The purpose of this worksheet is for you to familiarize yourself in tracing heap insertions and removal and its tree representations. For each problem, you should be drawing the tree representation for each heap operation. Then, you'll convert the tree representations into array representations, and then <u>**hardcode**</u> **the answers in Worksheet.java**. **You don't need to submit the tree representations,** but they are effective visualizations for you to trace heap structure. **Only answers in Worksheet.java will be graded.**

1. Insert the following elements in the given order to an empty **binary (d = 2) max-heap**. Draw out the tree representation of the heap after all elements are inserted. Then, store the array representation of the heap in variable **output** located within the method **insertionResult**.

   Elements to insert: [43, 12, 37, 44, 83, 13, 68, 59, 94, 71, 99, 3]

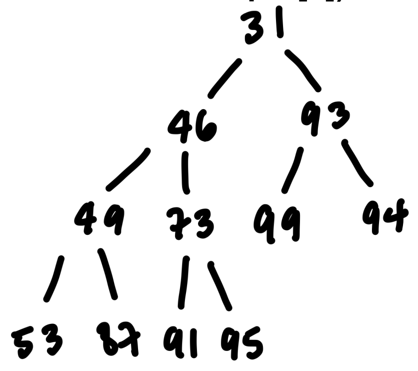


2. Remove the top element 5 times from the given **binary min-heap** and draw the tree representations of the initial heap and the heap after **each** removal. Then, for each iteration of removal, convert your tree representation into array representation. Store the values into the method **removalResult**.

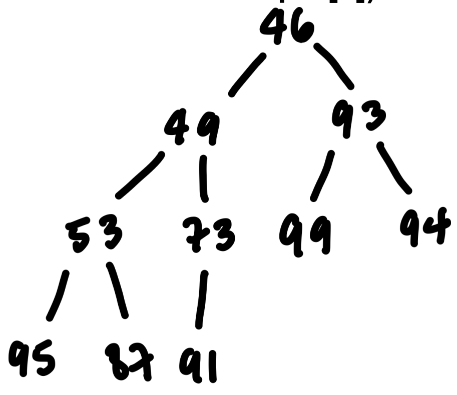
   Array representation of the initial heap: [30, 46, 31, 49, 73, 93, 94, 53, 87, 91, 95, 99]

   Note: Within Worksheet.java, you're given an empty 5-by-12 2D array **output**. For row `i`, you'll be storing the array representation of the `i+1`th removal. If the number of elements within the heap is less than 12 elements, then you pad it with 0's at the back. **Eg**: if the array representation after 1st removal is [46, 39, 100, 93, 96, 101, 64, 20, 34, 30, 91], then output[0] = [46, 39, 100, 93, 96, 101, 64, 20, 34, 30, 91,0]

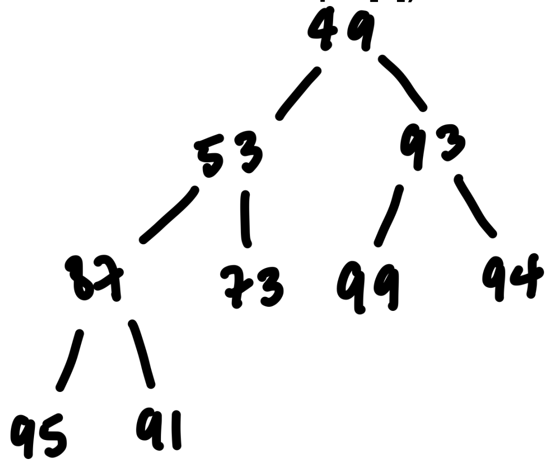
After 1st removal (Answer stored in output[0])



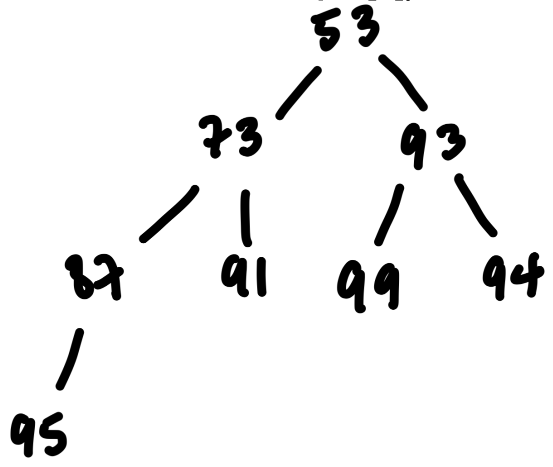
After 2nd removal (Answer stored in output[1])



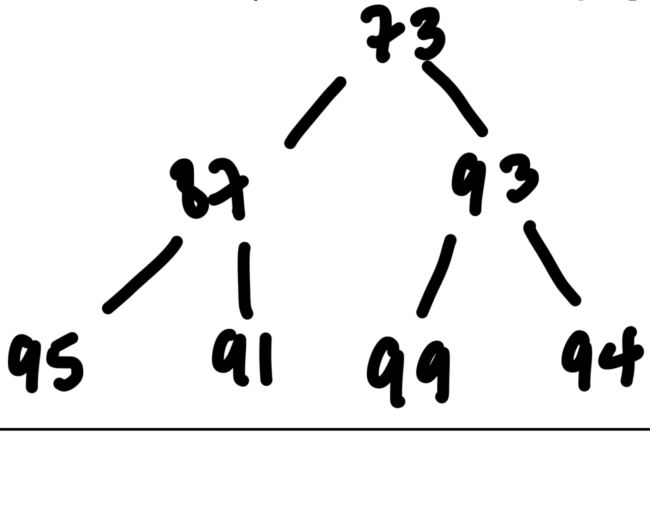
After 3rd removal (Answer stored in output[2])



After 4th removal (Answer stored in output[3])



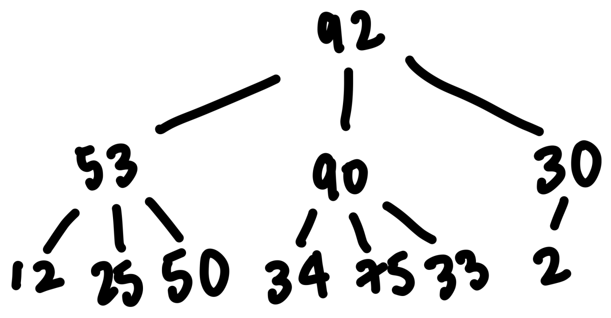
After 5th removal (Answer stored in output[4])



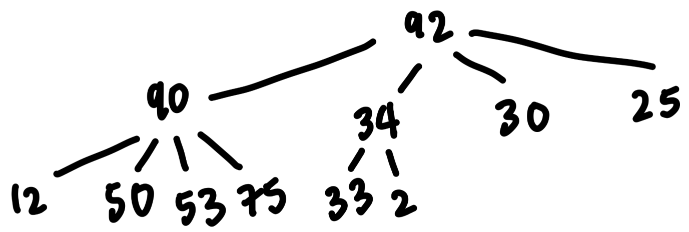
3. Draw the tree representations of the d-ary max-heaps from the following order of insertion. Then, convert the tree representations to array representations and store your result in variable **output**, located within the method **dResult**. You'll be performing the above process for d = {3, 4}. The array representation of d=3 will be stored in output[0], and the array representation of d=4 will be stored in output[1]

   Elements to insert: [12, 50, 34, 30, 25, 90, 53, 75, 92, 33, 2]

3-ary



4-ary



4. Store the array representations of the given **3-ary max-heap** after each specified operation into 2d array **output**, located within the method **heapOperations**. You'll be storing the array representation at the `i`th series of operations at row `i-1`. Variable output has 12 columns, given that the maximum number of elements under the series of operations is 12. If the array representation after an operation is less than 12, you pad 0's at the end(See Q2 for more details)

   Note that **each operation should be performed on the result of the previous operation**.

Original											
73	42	71	19	27	41	12	53	58	9	12	
Operation 1: After **removing the top twice** (Store at index 0)											
58	42	53	19	27	41	12	9	12			
Operation 2: After **inserting 10 and 90** (Store at index 1)											
90	42	53	58	27	41	12	9	12	10	19	
Operation 3: After **removing the top once** (Store at index 2)											
58	42	53	19	27	41	12	9	12	10		
Operation 4: After **inserting 67 and 83** (Store at index 3)											
83	42	53	67	27	41	12	9	12	10	19	58
Operation 5: After **removing the top 5 times** (Store at index 4)											
41	27	12	19	9	10	12					