**LAB & TUT 7 – Heap**

**PART 1 – LAB**

1. **Complete the flowing functions: InsertHeap, DeleteHeap, BuildHeap**
2. **Heap property checking:** write a function to check if an array is a max heap:

bool IsMaxHeap(int \*arr, int size)

1. **Delete an arbitrary node:** The provided **deleteHeap** function is only able to delete the root of a heap. Write another function that allows you to delete any node in a heap:

bool DeleteHeapNode(int \*&maxHeap, int delPosition,

int &size, int & dataOut)

1. **Printing a heap as a normal tree:** Write a program that prints a heap as a normal tree. For example, with the heap:

int size = 31;

int \*maxHeap = new int[size];

for (int i = 0; i < size; i++) { maxHeap[i] = i; }

buildHeap(maxHeap, size);

Your program should print the following output:



Here is another example



**Note:** for printing purpose, we assume that our heap only consists of integers ranged from 0 to 999 or else there will be displacement problem. If a node is less than 100, you will have to pad zeros in front of it so that it has exactly tree characters

**PART 2 – TUT**

1. **Building a heap:**

Given the following arrays:

int maxHeap1[8] = {56, 45, 4, 77, 60, 34, 35, 22};

int maxHeap2[9] = { 1, 3, 5, 7, 9, 2, 4, 6, 8};

Build a max heap for each of them. Draw the final result.

1. **Delete heap:**

After finishing building the two heaps above, apply the **DeleteHeap** operation on each of them **twice**. Draw the result.