ALGORITHMS AND DATA STRUCTURES 2

TOPIC 9: HEAPS
FORMATIVE EXERCISE

In this activity, you will work **individually** to implement a binary heap.

PART 1: YOUR TASK

In this formative exercise, your task is to implement a binary heap data structure and the heapsort algorithm.

This exercise will be automatically graded. Thus, you are provided with two files: Heap.cpp and Heap.hpp.

Your task is to complete the code that makes the methods in that skeleton code operative. **You must not change the prototypes of the methods**. You can add other methods and variables if you want.

PART 2: THE METHODS

Please look at the content of the skeleton files you are provided with for this formative exercise (Heap.cpp and Heap.hpp). You can see there are eleven methods you must implement in the .cpp file:

- parent(index): This method returns the index of the parent of the element stored in index.

- left(index): This method returns the index of the left child of the element stored in index.

- right(index): This method returns the index of the right child of the element stored in index.

- Heap(n): This is a simple constructor with a single integer input argument. The constructor creates the storage area for a heap of size *n* and initialises the heap size.

- Heap(list, bool): This constructor received two input arguments: a list and a Boolean value. If the Boolean argument is TRUE, the constructor must build the heap incrementally from the list (using the method insert, described below). Otherwise, it must build the heap in place (using the method buildMaxHeap, described below). In both cases, make sure the constructor creates the storage area for the heap and initialises the heap size correctly.

- insert(k): This method inserts element k in the heap (maintaining the heap properties).

- maximum(): This method returns the element with the maximum value in the heap (it does not extract it).

- maxHeapify(index): This method applies max-heapify from the element in position index ‘downwards’ (in the tree visualisation).

- buildMaxHeap(): This method builds the heap in place.

- extractMax(): This method extracts the maximum element from the heap (maintaining the heap properties) and returns its value.

- sort(): This method uses Heapsort to sort the content of ‘array’. It returns the sorted array.

PART 3: SUBMISSION

When you want to submit your work, please compress both files (.cpp and .hpp) into a zip file. The compressed file can have any name.

In the My submission tab, press the blue ‘Create Submission’ button. Below the text ‘Upload Files and Submit’, press the blue ‘Heaps’ button, select the file and click ‘Submit’.

Upon submitting, you will see the date of the submission in grey. It might take a while for the system to grade your code. You can wait for the grade or log out and come back later.

Once the grade is ready, you can click on the submission tab and then on ‘Show grader output’ to check on eventual errors.

You can upload your submission as many times as you want. Your highest score will be recorded.

You can get familiar with the system by simply uploading the skeleton code provided for you. Naturally, all tests will fail in this case.

Next, attempt to implement the methods. The following order is recommended:

- the parent / left-child / right-child helper routines
- the simple constructor
- insert
- maximum

- the incremental constructor
- maxHeapify
- builMaxHeap
- extractMax
- sort.

Note that you’ll get NullPointer types of errors when the constructors are not ready. After each step, run the tests once more, and check that the number of failing tests has decreased.