ALGORITHMS & DATA STRUCTURES
Lahcen Ouarbya

BINARY HEAPS
LAB. SUBMISSION

In this activity, you will work **individually** implementing a binary heap data structure and the Heapsort algorithm. Remember: you can discuss ideas with your classmates, but you cannot see other's work neither others can see your individual work.

After finishing the implementation of your binary heap and the Heapsort algorithm, you must upload **a single file** with your code using the "Heaps Lab Submission" link available at learn.gold, Week 5. Your code will be graded immediately.

PART 1: YOUR TASK

In this lab submission, your task is implementing a binary heap data structure and the Heapsort algorithm.

Because this lab submission will be automatically graded, you are provided with a skeleton code:

- a single file (Heap.java) for Java programmers
- two files (Heap.cpp and Heap.hpp) for C++ programmers.

Your task is completing the code that makes the methods in that skeleton code operative. **You must not change the prototypes of the methods**. If you do that, your code will not pass the automatic tests and you will not get a good mark. You can add other methods if you want.

PART 2a: JAVA PROGRAMMERS

Please, look at the content of the skeleton file you are provided for this lab submission (Heap.java). You can see 10 signatures of methods you must implement:

CONSTRUCTORS

- Heap(int size): It creates the array for a heap of size n and updates the variable heap_size

- Heap(List<Integer> source, boolean incremental): If variable 'incremental' is TRUE, it builds the heap incrementally from the data stored in the list (using the method insert). Otherwise, builds the heap in-place (using the method buildMaxHeap).

Note: You'll get NullPointer type of errors from the auto-grader while the constructors are not ready.

'MOVEMENT' OPERATIONS

- int parent(int index): it returns the index of the parent of the element stored in index
- int left(int index): it returns the index of left child of the element stored in index
- int right(int index): it returns the index of right child of the element stored in index

HEAP OPERATIONS

- void maxHeapify(int i): it applies max-heapify from element in position index "downwards" (in the visual representation of the heap)
- void buildMaxHeap(): it builds a heap in place
- void insert(Integer k): it inserts element k in the heap (maintaining heap properties)
- Integer maximum(): it returns the value of the element with the maximum value in the heap (it does not extract it)
- Integer extractMax(): it extracts the maximum element from the heap (maintaining the heap properties) and returns its value

Besides the above methods, you must implement heapsort. For that, use the following signature:

- ArrayList<Integer> sort(): it sorts the content of 'array' and returns the array sorted using heapsort.

PART 2b: C++ PROGRAMMERS

Please, look at the content of the skeleton files you are provided for this formative exercise (Heap.cpp & Heap.hpp). You can see there are 10 methods you must implement in the .cpp file:

'MOVEMENT' OPERATIONS

- 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

CONSTRUCTORS

- Heap(n): This is a simple constructor with a single int 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 method buildMaxHeap, describe below).

Note: You'll get NullPointer type of errors from the auto-grader while the constructors are not ready.

HEAP OPERATIONS

- 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 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

Besides the above methods, you must implement heapsort. For that, use the following signature:

- std::vector<int> Heap::sort(): it uses Heapsort to sort the content of array. It returns the sorted array.

PART 3: SUBMISSION

For this lab submission your code will automatically graded. As soon as you upload your submission, your code will be analysed and a report will be displayed on the screen. The report includes your mark in the assignment as well as any errors you still might need to correct.

You can upload your submission as many times as you want, without any block window. Your highest score will be considered.

You can get familiar with the system simply uploading the skeleton code provided to you. In that case, you will get a mark of 39. To do so, you must:

- select the language of your submission (Question 1)

- upload the file (Question 2). In the case of C++ programmers, you must compress the .cpp and .hpp files and upload the resulting .zip file
- answer the last personalised questions (Questions 3 & 4). You can omit these questions. In that case, you will not get full marks, but you can still check your code).

As soon as you press the button "Submit" you will see the message "Verifying your answers….".

Because none of the methods is implemented in the skeleton code yet, you will be next shown a red screen (signalising a mark under 40) with the following message: "There are some errors in your answer. Your score is 39%".

Below the red screen, you will receive a report with the tests that your code failed.

After implementing each method, submit again and check that the number of failing tests has decreased.