CS 15: Heaps

Introduction

In this lab, you will implement some functions of a `Heap` class. The `Heap` for our lab will be a min heap of strings that is implemented with an array. The provided `main` driver will use these functions in order to sort the elements in an array of strings using an instance of the `Heap`. Once the array is sorted, a secret message will be printed out!

Starter Code

Heap Class

Much of the `Heap` has been implemented for you already. The header file is complete- check there for some useful comments! In the `heap.cpp` file, you will find that the constructors, `insert` (which you won't use in this lab), and `size` functions are all already implemented. An `ensureCapacity` function, used by `insert`, is also written for you. Also, helper functions to access the right/left children or parent indices of a given index are implemented for you.

Driver

The driver, in `main.cpp`, is written for you. It builds a heap, sorts the elements with repeated calls to `removeMin()`, and prints the results. Note that we will discuss a better way to do this sorting algorithm in class — to do it "properly" requires private access to the heap and destroys the data structure, but is more space efficient. However, the idea behind the lab is for you to write various functions of the `Heap`.

Code to Write

You will be responsible for implementing the following functions:

- `Heap::~Heap()`
  Heap destructor.
- `std::string Heap::min()`
  Returns the minimum element in the heap.
- `void Heap::downHeap(int location)`
  This function repeatedly swaps a node with its smallest child until the heap property is restored. Remember: all children are bigger than their parent.
- `std::string Heap::removeMin()`
  This function is exactly as discussed in class: the root value will be returned, the heap size shrinks by one, and the value in the rightmost node of the last level of the tree is copied to the root. Then, down heaping begins from the root to reestablish the heap invariants.
- `void Heap::buildHeap()`
  This function will start halfway through the list (array) and repeatedly call `downHeap` on the first half of the list. It is used by the non-default constructor, which takes in a list of strings and builds a heap from the list.
- (optional) `void Heap::upHeap(int location)`
  Fixes heap invariant by 'bubbling up' from the current index.

Getting Started

First, make a directory for this lab, change into that directory, and copy the lab files from the usual place. Next, look in the `Makefile`. You'll see the name of the executable program it builds.

Submitting

You will need to submit the following files:

```
main.cpp
heap.cpp
```

```
heap.h
README
Makefile
```

You must submit them using Gradescope to the assignment `lab_heaps`. Submit these files directly, do not try to submit them in a folder.