# California State University, Fresno

# DEPARTMENT OF COMPUTER SCIENCE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Class: | **Algorithms & Data Structures** | | | Semester: | **Fall 2021** |
|  | | | | | |
| Points |  | Document author: | **Marlon Branham** | | |
|  | Author’s email: | **zekebranham@mail.fresnostate.edu** | | |
| Laboratory number: | **Laboratory 2** | | |
|  | | | | | |

**1. Statement of Objectives**

In this lab, we were to develop a program that would create a heap data structure (both max and min) and provide functionality that builds a heap, maintains heap structure, can insert values, extract min/max values, and modify values. The significance of this lab is the ability to work with heap structures and be able to access, modify, and gather information from them. These were all accomplished in this lab. This lab report will cover my experimental procedures, analysis, encountered problems, and a conclusion of my results.

**2. Experimental Procedure**

We were given the task to create both a maximum-heap, and a minimum-heap. In both, using functions to work with them. We were provided with a header file with all the necessary functions we needed to complete. For both max-heap and min-heap, we were to create the following functions: modifyVal, insertVal, heapify, extractMin/Max, accendingHeapSort, decendingHeapSort, and buildHeap. Totaling at 6 functions per heap, 12 functions total, majority of them were fairly straightforward and do not require much explanation beyond their titles. The only one that was altered was the extractMin/Max function. This function itself is very simple and straightforward, but in order to replicate the results in the given SampleOutput file, it made it quite difficult. Since the extract functions take in a given array/heap, swap the first and last value, then reduce the size of the heap with heapify, it made it difficult when the ascending/descending functions change the structure of the array. With that, I found my way around this by using two functions, one that returns the max/min separately, then the other to extract the min/max.

A computer screen shot of a black background with white text

Description automatically generatedA computer screen shot of white and blue text

Description automatically generated

**3. Analysis**

My results replicated the SampleOutput file the best I could. I included an extra statement that displayed the heap prior to sorting. I created the choice for the user to pick ascending or descending order using 0 for ascending and 1 for descending. I displayed the original heap, the sorted heap, the inserted extra element, the extracted maximum, and the final sorted heap. The results were identical for the descending option.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

**4. Encountered Problems**

There were a few encountered problems with this lab. No problems with the actual functionality of the functions themselves, but issues arose when the sorting algorithms were introduced. The reason they caused problems were because it affected the workings of the extractMin/Max functions. For example, speaking in terms of the max-heap portion with ascendingHeapSort, when the heap was sorted in ascending order, since the extractMax function will swap the first(root) and last(leaf) elements and reduce the size of the heap by 1 before heapsorting, instead of swapping the max value with the smallest value(leaf) then removing the max value by reducing the size of the heap, ascending order caused the leading smallest value(leaf) to be swapped with the maximum at the end of the array, then would delete the smallest value instead of the maximum. To visualize, a heap example in this situation would be: “9, 6, 2, 3, 4” then would be sorted in ascending order: “2, 3, 4, 6, 9” then extractMax would be called which would swap the first and last element then reducing the size by one leaving: “3, 4, 6, 9.” In turn, to resolve this issue, I created sorted the array in descending order before the extractMax function was called to get the correct value.

As well as this, in order to produce the SampleOutput file exactly, it was impossible to have an *int extractMax* function because it would overlap with my sorting calls due to the SampleOutput wanting the full sorted array prior to the call of extractMax. Since Arr[0] would not be the maximum anymore after sorting in ascending order, changes would need to be made. This was resolved by using two functions: *int maximum* and *void extractMax.* This allowed me to find and store the maximum into an integer from Arr[0] called once the heap is built prior to sorting, then call the function to truly extractMax after the sorting occurred. Again, this was only an issue because of me trying to replicate the given results exactly. This would not be an issue in different circumstances and could use *int extractMax* instead of the two *int maximum* and *void extractMax.*

A computer screen shot of a program

Description automatically generated

**5. Conclusions**

What I learned in this lab was that heaps are very tricky to deal with sometimes. In regard to their basic function, they are relatively straightforward and easy to understand. But working with multiple different methods of altering and modifying heaps makes working with them a bit more difficult, in specific sorting. Overall, a successful lab assignment, learning a deeper understanding of heaps and how to work with them.

**6. References**

None.