CALIFORNIA STATE UNIVERSITY, FRESNO
DEPARTMENT OF COMPUTER SCIENCE

Class:	**Algorithms & Data Structures**	Semester:	**Fall 2023**

Points		Document author:	**Pratham Aggarwal**
		Author's email:	**Pratham_aggarwal@mail.fresnostate.edu email**
		Laboratory number:	**Lab 4**

1. Statement of Objectives

The main goal of this lab is to provide a thorough knowledge and analysis of a C++ program that implements the two important max heaps and min heaps data structures. These data structures are crucial to many algorithms and applications and are basic to computer science.

Insertion of Elements: The user may enter a number of elements, which, depending on the user's preference, will be arranged into either a max heap or a min heap.
Value Modification: The heap data structure's values are modifiable by users. By using this capability, items can be updated while still maintaining the heap's basic structure and features.
Extraction of Extreme Values: Depending on whether the heap is a max heap or a min heap, the program allows the extraction of extreme values, which can be either the maximum or least value in the heap. Several algorithms and data processing activities require this operation.
Operations for Sorting: This algorithm provides the ability to either sort the components of the heap in ascending or descending order.

The importance of this lab is in the practical experience with heap data structures that it provides. We can immediately relate to these structures and study them, which helps us better grasp how heaps function and how they can be used in practical applications.

2. Experimental Procedure

It is the algorithm to work with max and min heaps. The process is broken down into the following steps:

Instances of the Heap class are created first. These instances serve as both the max and min heaps, giving the elements a container for organization and manipulation.

User input: The number of elements the user desires to work with is the first thing we take input. Second, the elements themselves are taken as input. The program will process this user input as its initial dataset.

Selecting a Sorting Order: The user can select an ascending or descending sorting order on how they want the components of the heap are arranged and presented.

Execution of Operation: Depending on the sorting method selected (ascending or descending), the algorithm runs accordingly. These operations include building the heap, inserting elements, sorting the heap, and extracting extreme values (maximum or minimum). Based on user input and the desired sorting order, different actions are carried out.

3. Analysis

Upon analyzing the program's output against the provided expected outputs, I am encountering some issues with the program's logic. Below are both output scenarios in detail:

Output 1 (Ascending Order):

Input: 4 3 2 6
Sorting order: Ascending
Input element: 9
Expected sorted heap: 2 3 4 6 9
Actual sorted heap (program output): 2 4 3 6 9 (incorrect)
Extracted maximum: 6 (which is incorrect)
Sorted heap after extraction: 2 3 4 6 9 (which is incorrect)

Output 2 (Descending Order):
Input: 4 3 2 6
Sorting order: Descending
Input element: 9
Expected sorted heap: 9 6 4 3 2
Actual sorted heap (program output): 9 6 2 3 4 (incorrect)
Extracted minimum: 2 (which is correct)
Sorted heap after extraction: 9 6 2 3 4(which is incorrect)

Screenshot attached in the end of the report.

4. Encountered Problems

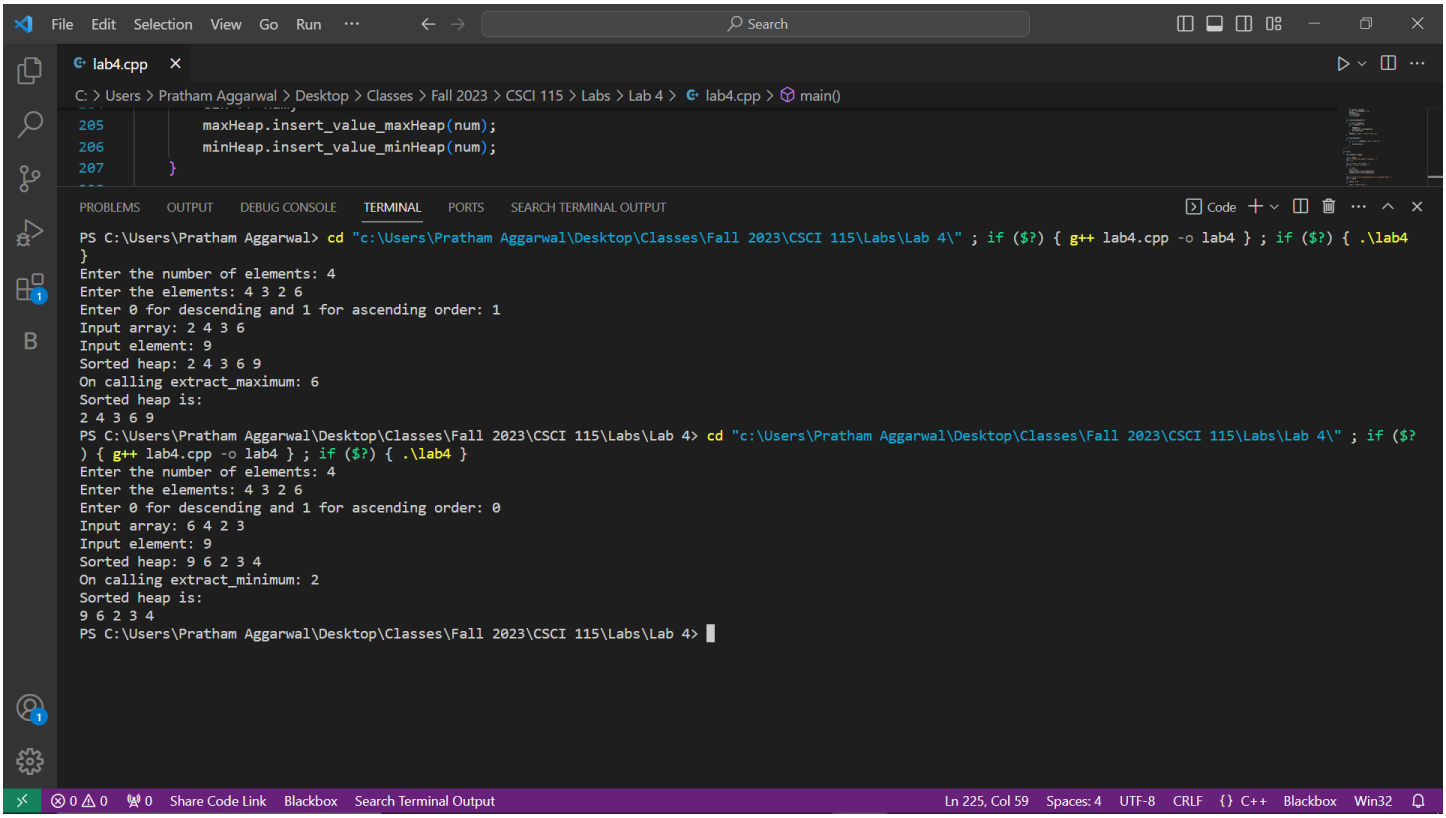
Firstly, I had to look up some functions algorithms because I was not getting correct output, but I got them fixed as it was just basic error. One problem that I am encountering is that the program generates inaccurate results for both scenarios of ascending and descending order. The program's generated sorted heaps in both situations don't correspond to the outcomes that were anticipated. I think there is some logical flaw. Probably, the issue lies in the sorting of the heap after inserting an additional element. The program's logic for sorting is not correctly maintaining the max and min heap properties.

5. Conclusions

In conclusion, the algorithm was supposed to implement max and min heap operations is currently providing inaccurate results. After adding a new element, the algorithm fails to sort the heap effectively, which causes differences between the intended and actual outputs. To make sure the application works as planned, more debugging and sorting logic correction are needed. However, this lab has given a useful understanding of heap data structures and how they work, emphasizing how crucial it is to preserve the proper heap characteristics across insertions and extractions.

6. References

N/A


File Edit Selection View Go Run
Search
lab4.cpp
C: > Users > Pratham Aggarwal > Desktop > Classes > Fall 2023 > CSCI 115 > Labs > Lab 4 > lab4.cpp > main()
205 maxHeap.insert_value_maxHeap(num);
206 minHeap.insert_value_minHeap(num);
207 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH TERMINAL OUTPUT
Code
PS C:\Users\Pratham Aggarwal> cd "c:\Users\Pratham Aggarwal\Desktop\Classes\Fall 2023\CSCI 115\Labs\Lab 4\" ; if ($?) { g++ lab4.cpp -o lab4 } ; if ($?) { .\lab4
}
Enter the number of elements: 4
Enter the elements: 4 3 2 6
Enter 0 for descending and 1 for ascending order: 1
Input array: 2 4 3 6
Input element: 9
Sorted heap: 2 4 3 6 9
On calling extract_maximum: 6
Sorted heap is:
2 4 3 6 9
PS C:\Users\Pratham Aggarwal\Desktop\Classes\Fall 2023\CSCI 115\Labs\Lab 4> cd "c:\Users\Pratham Aggarwal\Desktop\Classes\Fall 2023\CSCI 115\Labs\Lab 4\" ; if ($?
) { g++ lab4.cpp -o lab4 } ; if ($?) { .\lab4 }
Enter the number of elements: 4
Enter the elements: 4 3 2 6
Enter 0 for descending and 1 for ascending order: 0
Input array: 6 4 2 3
Input element: 9
Sorted heap: 9 6 2 3 4
On calling extract_minimum: 2
Sorted heap is:
9 6 2 3 4
PS C:\Users\Pratham Aggarwal\Desktop\Classes\Fall 2023\CSCI 115\Labs\Lab 4>
Share Code Link Blackbox Search Terminal Output
Ln 225, Col 59 Spaces: 4 UTF-8 CRLF C++ Blackbox Win32