Instructions: Please read carefully

- Please rename this file as only your ID number **(e.g. 18-*****-1.doc or 18-*****-1.pdf).**

Question

Implement the following for a max heap tree

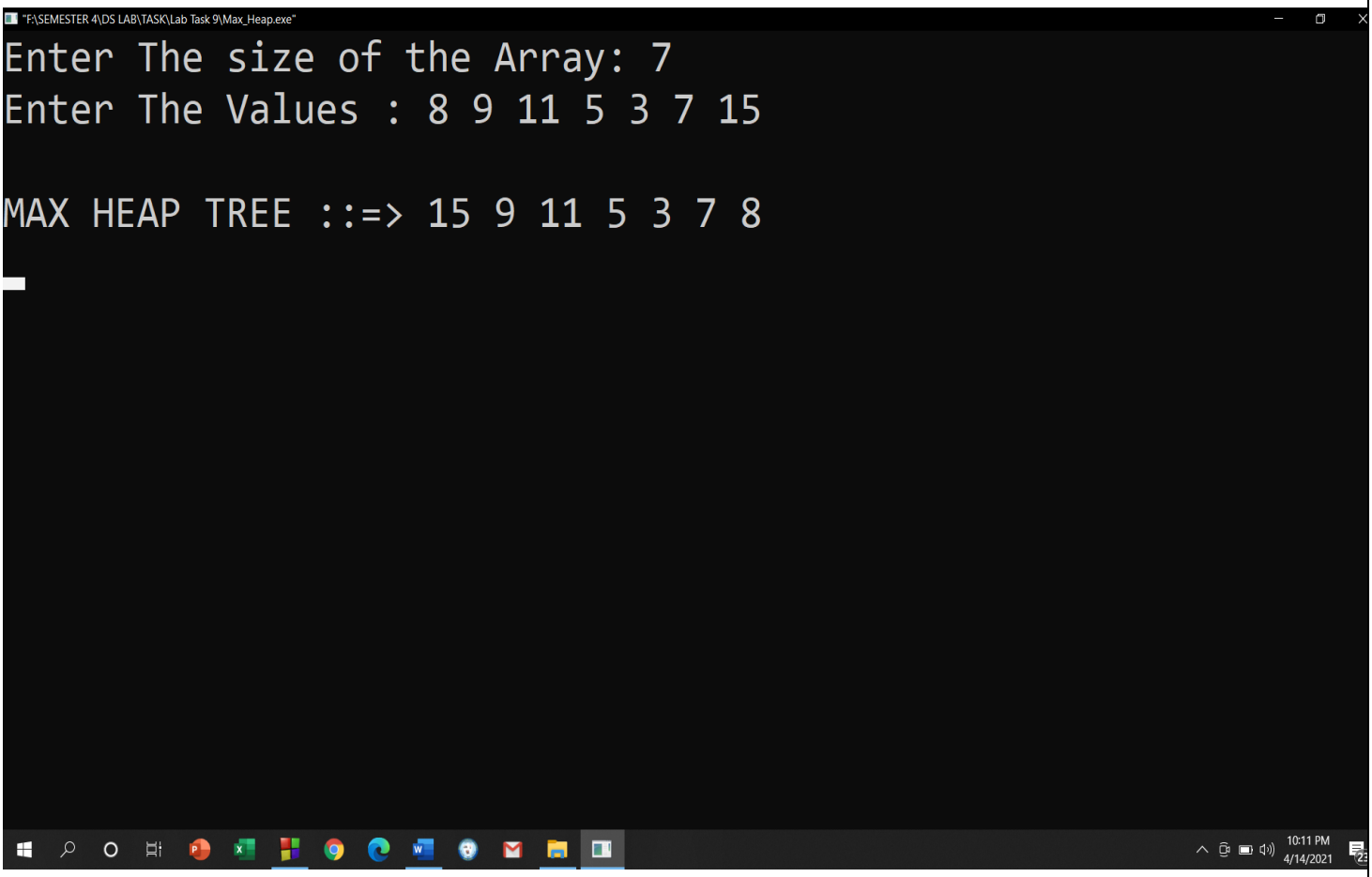
- Insertion
- Heapify
- Deletion

Your code here:

```
#include<iostream>
#include<conio.h>
using namespace std;

void heapify(int arr[],int  n,int i)
{

 int largest = i;
   int l = 2 * i + 1;
   int r = 2 * i + 2;

   if (l < n && arr[l] > arr[largest])
      largest = l;

   if (r < n && arr[r] > arr[largest])
      largest = r;

   if (largest != i) {
      swap(arr[i], arr[largest]);
      heapify(arr, n, largest);
   }

}
void MaxHeap(int arr[],int n)
{
   for(int i=n/2-1;i>=0;i--)
   {
      heapify(arr,n,i);
   }
}

void Insert(int arr[], int n)
{
   cout<<"Enter The Values : ";
   for(int i=0;i<n;i++)
         cin>>arr[i];
}
void Display(int arr[], int n)
{
             for(int i=0;i<n;i++)
      cout << arr[i] << " ";
   cout << endl;
}
```

```
int main()
{
        int n;

        cout<<"Enter The size of the Array: ";
        cin>>n;
  int arr[n];
  Insert(arr,n);
  MaxHeap(arr,n);
        cout << "\nMAX HEAP TREE ::=> ";
        Display(arr, n);
        getch();
}
```

Your whole Screenshot here: (Console Output):



Pseudocode	

Heapify	Heapify(array, size, i)
	  set i as largest
	  leftChild = 2i + 1
	  rightChild = 2i + 2
	
	  if leftChild > array[largest]
	  set leftChildIndex as largest
	  if rightChild > array[largest]
	  set rightChildIndex as largest
	
	  swap array[i] and array[largest]
	
	//code
	void heapify(vector<int> &hT, int i)
	{
	  int size = hT.size();
	  int largest = i;
	  int l = 2 * i + 1;
	  int r = 2 * i + 2;
	  if (l < size && hT[l] > hT[largest])
	  largest = l;
	  if (r < size && hT[r] > hT[largest])
	  largest = r;
	
	  if (largest != i)
	  {
	  swap(&hT[i], &hT[largest]);
	  heapify(hT, largest);
	  }
	}

Create a Max Heap	MaxHeap(array, size)
	  loop from the first index of non-leaf node down to zero
	  call heapify

Insert	If there is no node,
	  create a newNode.
	else (a node is already present)
	  insert the newNode at the end (last node from left to right.)
	
	heapify the array

Delete	If nodeToBeDeleted is the leafNode
	  remove the node
	Else swap nodeToBeDeleted with the lastLeafNode
	  remove noteToBeDeleted
	
	heapify the array