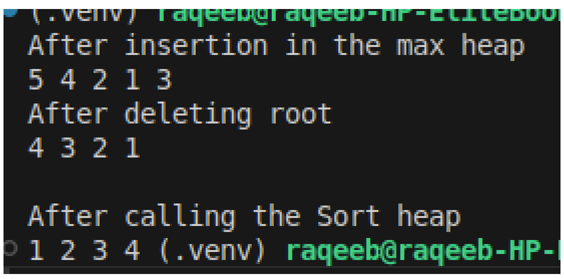
Hashing

```
#include<iostream>

using namespace std;

class Heap
{
private:
int *My_array;
int size;
int length;
public:

// Constructor to initialize the heap with a given size
Heap(int size)
{
this->size=size;
My_array= new int(size);
length=0;
}


// Function to insert a value into the max heap
void Heap_insertion_max_heap(int value)
{
if(size==length)
{
cout<<"The heap is full"<<endl;
return;
}

My_array[length]=value;
heapify_up_max(length);
length++;
}


// Function to restore the max heap property by
heapifying up
void heapify_up_max(int index)
{
if(index<=0)
return;
int parent=(index-1)/2;
if(My_array[index]>My_array[parent])
{
swap(My_array[index],My_array[parent]);
heapify_up_max(parent);
}

}


void heap_display()
{
if(length==0)
{
cout<<"the heap is empty"<<endl;
return;
}

for(int i=0;i<length;i++)
{
cout<<My_array[i]<<" ";
}
}
```

```
// Function to remove and return the root (maximum
element) from the heap
void deleteRoot()
{
if(length==0)
{
cout<<"The heap is empty"<<endl;
return;
}
My_array[0]=My_array[length-1];
length--;
heapify_down_max(0);

}


// Function to restore the max heap property by
heapifying down
void heapify_down_max(int index)
{
int left_side = 2*index + 1;
int right_side = 2*index + 2;
int largest = index;

if(left_side < length && My_array[left_side] >
My_array[largest])
{
largest = left_side;
}
if(right_side < length && My_array[right_side] >
My_array[largest])
{
largest = right_side;
}
if(largest != index)
```

```
{
swap(My_array[index], My_array[largest]);
heapify_down_max(largest);
}
}


// Function to sort the elements in the heap in
ascending order using heap sort
void heap_sort()
{
if(length==0)
{
cout<<"The heap is empty"<<endl;
return;
}

int* tempArray = new int[length];
int tempLength = length;

for(int i = 0; i < tempLength; i++)
{
tempArray[i] = My_array[i];
}

for(int i = tempLength - 1; i >= 0; i--)
{
swap(My_array[0], My_array[length - 1]);
length--;
heapify_down_max(0);
}



delete[] tempArray;
length = tempLength;
```

```
}

};


int main(void)
{
Heap obj(5);
obj.Heap_insertion_max_heap(1);
obj.Heap_insertion_max_heap(2);
obj.Heap_insertion_max_heap(3);
obj.Heap_insertion_max_heap(4);
obj.Heap_insertion_max_heap(5);
cout<<"After insertion in the max heap"<<endl;
obj.heap_display();
cout<<endl;

obj.deleteRoot();
cout<<"After deleting root"<<endl;
obj.heap_display();
cout<<endl;


obj.heap_sort();
cout<<"\nAfter calling the Sort heap"<<endl;
obj.heap_display();

return 0;
}
```
output:



```
(.venv) raqeeb@raqeeb-HP-EliteBoo
After insertion in the max heap
5 4 2 1 3
After deleting root
4 3 2 1

After calling the Sort heap
1 2 3 4 (.venv) raqeeb@raqeeb-HP-
```