DSA_College\HeapSort.cpp

```cpp
// T.C. of Heap Sort is O(nlogn) in best and worst case both
// In heap sort, we first build the max Heap and delete from that heap


// Heapify method - apply to only non-leaf elements
// The leaf node would be from floor(n/2)+1 to n

#include<iostream>
using namespace std;

void maxHeapify(int A[],int n,int i)  // A is array, n is size of array, and i is the element from which we start
{  // T.C. = O(n)
   int large = i; // large is a non-leaf node
   int l = 2*i+1 ;
   int r = 2*i+2 ;
   while(l<n && A[l] > A[large])
   {
    large = l ;
   }
   while(r<n && A[r] > A[large])
   {
    large = r ;
   }
   if(i != large)
   {
    swap(A[large],A[i]) ;
    maxHeapify(A,n,large) ;
   }
}

void heapSort(int A[],int n) // T.C. = O(nlogn)
{
    for(int i=n/2-1;i>=0;i--)
    {
        maxHeapify(A,n,i) ;  // for building the max heap
    }
    // now we write deleting code, that is done from root element(top element of tree)
    for(int i=n-1;i>0;i--)
    {
        swap(A[0],A[i]) ;
        maxHeapify(A,i,0) ;
    }
}

void print(int arr[], int n)
{
    for(int i=0;i<n;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
```

```
52 }
53
54 int main()
55 {
56     int A[] = {23,66,12,4,81,40,100} ;
57     int n = sizeof(A)/sizeof(A[0]) ;
58     heapSort(A,n) ;
59     print(A,n) ;
60
61     return 0;
62 }
63
```