```
// C++ program for implementation of Heap Sort
#include <iostream>
using namespace std;

// To heapify a subtree rooted with node i which is
// an index in arr[]. n is size of heap
void heapify(int arr[], int n, int i)
{
	int largest = i; // Initialize largest as root Since we are using 0 based indexing
	int l = 2 * i + 1; // left = 2*i + 1
	int r = 2 * i + 2; // right = 2*i + 2

	// If left child is larger than root
	if (l < n && arr[l] > arr[largest])
		largest = l;

	// If right child is larger than largest so far
	if (r < n && arr[r] > arr[largest])
		largest = r;

	// If largest is not root
	if (largest != i) {
		swap(arr[i], arr[largest]);

		// Recursively heapify the affected sub-tree
		heapify(arr, n, largest);
	}
}
```

```
// main function to do heap sort
void heapSort(int arr[], int n)
{
	// Build heap (rearrange array)
	for (int i = n / 2 - 1; i >= 0; i--)
		heapify(arr, n, i);

	// One by one extract an element from heap
	for (int i = n - 1; i >= 0; i--) {
		// Move current root to end
		swap(arr[0], arr[i]);

		// call max heapify on the reduced heap
		heapify(arr, i, 0);
	}
}

/* A utility function to print array of size n */
void printArray(int arr[], int n)
{
	for (int i = 0; i < n; ++i)
		cout << arr[i] << " ";
	cout << "\n";
}

// Driver program
int main()
{
	int arr[] = { 60 ,20 ,40 ,70, 30, 10};
```

```
        int n = sizeof(arr) / sizeof(arr[0]);
//heapify algorithm
// the loop must go reverse you will get after analyzing manually
// (i=n/2 -1) because other nodes/ ele's are leaf nodes
// (i=n/2 -1) for 0 based indexing
// (i=n/2) for 1 based indexing
        for(int i=n/2 -1;i>=0;i--){
        heapify(arr,n,i);
}

cout << "After heapifying array is \n";
        printArray(arr, n);



        heapSort(arr, n);

        cout << "Sorted array is \n";
        printArray(arr, n);

return 0;
}
```