***Heap Sort:***

def heapify(arr, n, i):

largest\_index = i

left\_index = 2\*i + 1

right\_index = 2\*i + 2

if left\_index < n and arr[left\_index] > arr[largest\_index]:

largest\_index = left\_index

if right\_index < n and arr[right\_index] > arr[largest\_index]:

largest\_index = right\_index

if largest\_index != i:

arr[i], arr[largest\_index] = arr[largest\_index], arr[i]

heapify(arr, n, largest\_index)

def max\_heap(arr):

for i in range(int(len(arr)/2) - 1, -1, -1):

heapify(arr, len(arr), i)

print("Max Heap Tree")

for i in range(len(arr)):

print(arr[i], end=" ")

print()

def heap\_sort(arr):

for i in range(len(arr) - 1, -1, -1):

arr[0], arr[i] = arr[i], arr[0]

heapify(arr, i, 0)

def main():

arr = [1, 12, 9, 5, 6, 10]

print("Unsorted list")

print(arr)

max\_heap(arr)

heap\_sort(arr)

print("Sorted list")

print(arr) main()

Output:

Unsorted list

[1, 12, 9, 5, 6, 10]

Max Heap Tree

12 6 10 5 1 9

Sorted list

[1, 5, 6, 9, 10, 12]

***C Output:***

................max Heap tree............

12 6 10 5 1 9

................Sorted list............

1 5 6 9 10 12

C:

#include <stdio.h>

void swap(int \*x, int \*y)

{

int temp=\*x;

\*x=\*y;

\*y=temp;

}

void heapify(int a[],int n,int i)

{

int largestIndex=i;

int leftIndex=2\*i+1;

int rightIndex=2\*i+2;

if(leftIndex<n && a[leftIndex]>a[largestIndex])

{

largestIndex=leftIndex;

}

if(rightIndex<n && a[rightIndex]>a[largestIndex])

{

largestIndex=rightIndex;

}

if(largestIndex!=i)

{

swap(&a[i], &a[largestIndex]);

heapify(a,n,largestIndex);

}

}

void maxHeap(int a[],int n)

{

for(int i=n/2-1;i>=0;i--)

{ heapify(a,n,i);}

printf("................max Heap tree............\n");

for(int i=0;i<n;i++)

{

printf(" %d",a[i]);

}

}

void heapSort(int a[], int n)

{

for(int i=n-1;i>=0;i--)

{

swap(&a[0],&a[i]);

heapify(a,i,0);

}

}

int main()

{

int a[]={1,12,9,5,6,10};

int n=sizeof(a)/sizeof(int);

maxHeap(a,n);

heapSort(a,n);

printf("\n................Sorted list............\n");

for(int i=0;i<n;i++)

{

printf(" %d",a[i]);

}

return 0;

}