**AIM:-To perform heapsort for using array data structure.**

**THEORY:-**

* **A heap is a nearly complete binary tree with the following two properties:**
  + **Structural property: all levels are full, except possibly the last one, which is filled from left to right**
  + **Order (heap) property: for any node x**

**Parent(x) ≥ x**

**PROGRAM:-**

#include<stdio.h>

void maxheapify(int heap[],int newnode)

{

int temp,parent;

if(newnode>0)

{

parent=(newnode-1)/2;

if(heap[parent]<heap[newnode])

{

temp=heap[parent];

heap[parent]=heap[newnode];

heap[newnode]=temp;

}

maxheapify(heap,parent);

}

}

void heapdown(int heap[],int root,int lastindex)

{

int leftchild,rightchild,largechild,temp;

leftchild=(root\*2)+1;

rightchild=(root\*2)+2;

if(leftchild<=lastindex)

{//leftchild exists

if(rightchild<=lastindex)

{//both child exists

if(heap[leftchild]>heap[rightchild])

largechild=leftchild;

else

largechild=rightchild;

}

else

largechild=leftchild;

if(heap[root]<heap[largechild])

{

temp=heap[root];

heap[root]=heap[largechild];

heap[largechild]=temp;

heapdown(heap,largechild,lastindex);

}

}

}

void heapsort(int x[],int n)

{

int i,temp;

for(i=1;i<n;i++)

maxheapify(x,i);

printf("\n maxheapify array is\n");

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

printf("%d \t",x[i]);

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

{

temp=x[0];

x[0]=x[i];

x[i]=temp;

heapdown(x,0,i-1);

}

}

void main()

{

int n;

printf("enter no of elements \n");

scanf("%d",&n);

int x[n];

int i;

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

{

printf("enter the element %d\n",i+1);

scanf("%d",&x[i]);

}

printf("unsorted array is\n");

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

printf("%d \t",x[i]);

heapsort(x,n);

printf("\n sorted array is");

printf("\n");

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

{

printf("%d \t",x[i]);

}

}

**OUTPUT:-**

