S.No: 8	Exp. Name: ***Heap Sort***	Date:2025-09-16

Aim:

You're a software engineer at a financial services company responsible for developing algorithms to analyze market data and identify trading opportunities. Your team is exploring sorting techniques to efficiently process large datasets of stock prices. Your manager, David, provides the context:

As our trading platform handles a massive volume of market data, it's essential to implement efficient sorting algorithms to identify trends and trading opportunities quickly. One approach is to apply heap sort to arrange the stock prices in descending order, allowing traders to prioritize stocks with the highest potential returns. We need to develop a method to apply **heap sort** to the dataset and arrange the elements in **descending** order to support our trading strategy.

Constraints:

$1 <= N <= 1000$

$-100000 <= arr[i] <= 100000$

Use concept of heaps to sort the data.

Input Format:

The first line contains the integer N, the size of the list

The next line contains N space-separated integers.

Output Format:

Print the sorted list as a row of space-separated integers.

Source Code:

CTC29470.c

```
#include <stdio.h>
void heapify(int arr[], int n, int i){
   int la = i;
   int le = 2*i+1;
   int ri = 2*i+2;
   if(le<n && arr[le]>arr[la])
      la=le;
   if(ri<n && arr[ri]>arr[la])
      la=ri;
   if(la!=i){
      int t = arr[i];
      arr[i] = arr[la];
      arr[la] = t;
      heapify(arr,n,la);
   }
}
void heapsort(int arr[], int n){
   for(int i = n/2-1;i>=0;i--)
      heapify(arr,n,i);
   for(int i = n-1;i>=0;i--){
      int t = arr[0];
      arr[0] = arr[i];
      arr[i]=t;
      heapify(arr,i,0);
   }
}
```

```
void printd(int arr[],int n){
   for(int i = n-1; i>=0; i--){
      printf("%d ", arr[i]);
   }
   printf("\n");
}
int main(){
   int n;
   scanf("%d", &n);
   int arr[n];
   for(int i = 0; i<n; i++)
      scanf("%d", &arr[i]);
   heapsort(arr,n);
   printd(arr,n);
   return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
5
4 8 2 -3 0
8 4 2 0 -3

Test Case - 2
User Output
8
-6 -2 -8 -36 0 69 -5 -96
69 0 -2 -5 -6 -8 -36 -96