```
#include <stdio.h>


// swap function


void swap(int *a, int *b)
{
int temp = *b;
*b = *a;
*a = temp;
}


// Function to print the Heap as array


// will print as - 'message array[]\n'


void printArray(char message[], int arr[], int n)
{


printf("%s ",message);


for (int i = 0; i < n; ++i)
{
printf("%d ", arr[i]);
}


printf("\n");
}




// To heapify a subtree with node i as root


// Size of heap is n


void heapify(int arr[], int n, int i)
{
int largest = i; // Initialize largest as root
int leftChild = 2 * i + 1; // left child = 2*i + 1
int rightChild = 2 * i + 2; // right child = 2*i + 2


// If left child is greater than root


if (leftChild < n && arr[leftChild] > arr[largest])
largest = leftChild;


// If right child is greater than new largest


if (rightChild < n && arr[rightChild] > arr[largest])
largest = rightChild;
```

```
// If largest is not the root


if (largest != i)
{
// swap root with the new largest


swap(&arr[i], &arr[largest]);


// Recursively heapify the affected sub-tree i.e, subtree with root as largest
heapify(arr, n, largest);
}
}




// Function to build a Max-Heap from a given array


void buildHeap(int arr[], int n)
{
// Index of last non-leaf node
int lastNonLeafNode = (n / 2) - 1;


// Perform level order traversal in reverse from last non-leaf node to the root
node and heapify each node
for (int i = lastNonLeafNode; i >= 0; i--)
{
heapify(arr, n, i);
}
}




// Driver Code


void main()
{
// Array
int arr[] = {4, 18, 17, 10, 19, 20, 14, 8, 3, 12};


// Size of array
int n = sizeof(arr) / sizeof(arr[0]);


printArray("\nThe Array is : ", arr, n);


buildHeap(arr, n);
printArray("\nThe Array representation of Max_Heap is : ", arr, n);
}
```