```
def heapify(arr, n, i):
    # Initialize largest as root
    largest = i
    left = 2 * i + 1
    right = 2 * i + 2

    # Check if left child exists and is larger than the root
    if left < n and arr[left] > arr[largest]:
        largest = left

    # Check if right child exists and is larger than the largest so far
    if right < n and arr[right] > arr[largest]:
        largest = right

    # Change root if required
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]

        # Recursively heapify the affected sub-tree
        heapify(arr, n, largest)

def heap_sort(arr):
    n = len(arr)

    # Build a max heap
    for i in range(n//2 - 1, -1, -1):
        heapify(arr, n, i)

    # Extract elements one by one
    for i in range(n-1, 0, -1):
        # Swap current root with end
```

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

        # Heapify the reduced heap
        heapify(arr, i, 0)

def print_list(arr):
    for i in arr:
        print(i, end=" ")
    print()

# Driver segment to test the functions
if __name__ == "__main__":
    arr = [64, 34, 25, 12, 22, 11, 90]
    print("Unsorted list is:")
    print_list(arr)

    heap_sort(arr)

    print("\nList sorted in ascending order is:")
    print_list(arr)
```