```
print("ROHAN WAYAL")


# Heap Sort Implementation (using Max-Heap for ascending order sorting)
# Explanation:
# 1. Build a max-heap from the input array.
# 2. Repeatedly extract the maximum element from the heap (root of the heap),
#    swap it with the last element in the heap, and reduce the heap size by one.
# 3. Heapify the root again to maintain the max-heap property.
# 4. Continue until the heap size becomes 1.

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

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

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

    # If largest is not root, swap and continue heapifying
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)

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

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

    # Extract elements one by one from heap
    for i in range(n-1, 0, -1):
        # Move current root to end (since it's the largest)
        arr[i], arr[0] = arr[0], arr[i]
        # call max heapify on the reduced heap
        heapify(arr, i, 0)

    return arr



# Example usage
arr = [12, 11, 13, 5, 6, 7]
print("Original array:", arr)

sorted_arr = heap_sort(arr)
print("Sorted array:", sorted_arr)
```