Description. A Max-heap is an almost complete Binary tree satisfying the properties,

1. Every level but the last is full.
2. The bottom level is filled from the left up to some point.
3. $key(i) \leq key(parent(i))$ for every node i.

Note: A binary heap is typically represented as an array. The representation is done as:

- The root is at index 0 in array.
- Left child of i-th node is at (2*i + 1)th index.
- Right child of i-th node is at (2*i + 2)th index.

`Heapify` is the method of rearranging a heap to maintain the heap property. Using this a Max-heap can be built as follows.

Algorithm 1 Create_Max_Heap(A, n)

```
1: startIdx = (n/2) − 1;
2: for i = startIdx; i ≥ 0; i − − do
3:     HEAPIFY(A, n, i)
4: function HEAPIFY(A, n, i)
5:     largest = i;
6:     l = 2 × i + 1;
7:     r = 2 × i + 2;
8:     if l < n & A[l] > A[largest] then
9:         largest = l;
10:    if r < n & A[r] > A[largest] then
11:        largest = r;
12:    if largest ≠ i then
13:        SWAP(A[i], A[largest]);          ▷ Swap the array elements
14:        HEAPIFY(A, n, largest);
```

Example.

- **Input Array**: 4, 10, 3, 5, 1, 17, 15, 2, 21, 43, 100, 12.
  Output Max-Heap: 100, 43, 17, 21, 10, 12, 15, 2, 5, 4, 1, 3.
- **Input Array**: 1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17.
  Output Max-Heap: 17, 15, 13, 9, 6, 5, 10, 4, 8, 3, 1.

Write a complete MIPS-32 program satisfying the following requirements.

1. Reads an array of ten integers from the user. These numbers are collected from the input console using a loop and stored in the memory in an array called 'array'. **Do not store the numbers as scalars in ten different non-contiguous locations or in ten different registers.**
2. Write a recursive function ***heapify*** that applies the heapify property on an array.
3. Use the ***heapify*** function to create a max heap using Algo. 1.
4. Print the heap as the output of your program with suitable prompt.