Problema 2

```
def MayHeap(array):
    leng = len(array)
    verf = 0 #es el indice del right child
    i = 1 #indice del array
    h = 'y' #si es heap o no

    while verf < leng and h == 'y': #verifica si su child esta dentro del length del array y si
        verf = 2*i                  # al momento todavia es heap ('y')
        if array[i-1] < array[verf-1]:
            h = 'n'

        if verf < leng:
            if array[i-1] < array[verf]:
                h = 'n'

        i += 1

    if h == 'y':
        print 'Es Heap'
    else:
        print 'No es Heap'



possible_heap = [16,14,10,8,7,9,3,2,4,1]
MayHeap(possible_heap)
```

El running time del algoritmo es $O(lg(n))$

Problema 3

Algorithm 1 Max-Heapify

```
1:  procedure MAX-HEAPIFY(A, i)
2:      while u ≠ 0 do
3:          if Left(i) ≠ nil or Right(i) ≠ nil then
4:              u ← 1
5:              l ← Left(i)
6:              r ← Right(i)
7:              if l ≤ A.heap − size and A[l] > A[i] then
8:                  largest ← l
9:              else largest ← i
10:             if r ≤ A.heap − size and A[r] > A[largest] then
11:                 largest ← r
12:             if r ≠ i then
13:                 exchange A[i] with A[largest]
14:         else u ← 0
```