3)

a) Answer (A):
    Build Heap (A)
    return A[0]

Build Heap (A):
    for i in $\lfloor n/2 \rfloor - 1, \ldots, 0$:
        Shift Down (A, i)

Este algoritmo convierte el arreglo A en un Heap con O(n) y luego extrae la respuesta con mayor confianza con O(1).
A[n] = (respuesta, confianza)

Shift Down (A, i):
    if A[2i] v A[2i+1]:
        j = id (max (A[2i][1], A[2i+1][1]))
    if A[j][1] > A[i][1]
        A[j] ⇌ A[i]
        Shift Down (A, j)

b) next Answer (A):
    Build Heap (A) → Solo si no se ha inicializado Answer (A)
    A[0] ⇌ A[n-1]
    A.heap_size = A.heap_size - 1
    Shift Down (A, 0)
    best = A[0]
    A[0] ⇌ A[n-1]
    A[n-1] = ∅
    return best

El algoritmo intercambia la mejor respuesta por la peor y luego ordena el Heap, dejando la siguiente mejor en A[0], luego se guarda en una variable auxiliar para llevar al ultimo la segunda mejor respuesta y borrarla.

c) late Answer (A, respuesta, confiabilidad)
INSERT (A, (respuesta, confiabilidad))

INSERT (A, e)
i = primera celda vacia de A
H[i] = e
ShiftUp (A, i)

ShiftUp (A, i):
if A[⌊i/2⌋]:
j = ⌊i/2⌋
if H[j][1] < H[i][1]:
H[j] ⇌ H[i]
ShiftUp (A, j)

Este algoritmo aprovecha la funcionalidad de Heap para insertar la respuesta tardía. Esta, gracias a shiftup, llegará al inicio del arreglo si es la mas confiable.