Questão 1

(A) É uma lista de números inteiros e maiores que zero. A lista possui tamanho A.length e este tamanho é atribuído para a variável n.
↳ nº de itens na lista

Temos a variável "Key" que recebe o valor da primeira posição da lista. "j" é a posição do vetor. O "for" percorre do 2º ao "n" elementos da lista e compara o elemento na posição "j". Se o elemento da lista na posição "j" for menor que a variável armazenada em "Key", ela é armazenada na variável "Key" e o valor salvo anteriormente é subscrito.

(B) Loop invariante, pois o algoritmo faz operações em um array A[1...n] para j > 2 buscando o menor valor.

Questão 2 $n \geq 0$

$h(n) = \min\{f(n), g(n)\}$

$h(n) = f(n)$ se $f(n) \leq g(n)$

$h(n) = g(n)$ se $g(n) < f(n)$

$\min\{f(n), g(n)\} \leq f(n) + g(n)$

$\boxed{n \geq 0}$

(b)

- Escolho definir $f(n) = g(n) = n^2 = h(n)$

$f(n) + g(n) \in \Omega(h(n))$

$\boxed{\Omega(g(n)) \to f(n) \geq c\,g(n)}$

$n^2 + n^2 \geq c \cdot n^2$

$2n^2 \geq c \cdot n^2$

$\boxed{2 \geq c} \to$ para ser big omega

$\boxed{O(g) \to f(n) \leq c \cdot g(n)}$

$f(n) + g(n) \in O(h(n))$

$n^2 + n^2 \leq c \cdot n^2$

$\boxed{2 \leq c} \to$ para ser big-OH

- se eu escolher definir $f(n) \leq g(n)$, $h(n) = f(n)$

$f(n) = n = h(n)$ e $g(n) = n^2$

$n + n^2 \leq c \cdot n \to$ Big-OH

$n = 1 \to 2 \leq c \cdot 1 \to$ isso demonstra que não será Big OH nessa condição

- $n + n^2 \geq c \cdot n$

$n = 1 \to 2 \geq c \cdot 1 \to$ será Big omega p/ $c \leq 2$

- se eu escolher $g(n) < f(n)$, $h(n) = g(n)$

$g(n) = n$ e $f(n) = n^2$

temos o mesmo caso que $f(n) \leq g(n)$, $h(n) = f(n)$

(c) conforme demonstrações anteriores, será Big Oh de $h(n)$ se $f(n) = g(n) = h(n)$

(A) omega de $h(n)$ é o limite inferior de $f(n) + g(n)$

Questão 5

(a) A operação PUSH, em P, esta relacionado a quantidade de operações "De queue" que pode ser realizado.

(b) Push e Pop tem o custo de operação constante. Podemos escolher o custo da operação "Dequeue" 0 e a "Enqueue" = 4. Desta forma a "Enqueue" terá o custo de ambas operações, facilitando a contagem do custo do algoritmo. Ao realizar dessa forma, notamos que no fim da contagem a "Enqueue" O(n) ficara com um credito de operação (custo não usado na operação).

Questão 3:

ⓐ $T(n) = 3T(n-2)+1$

$T(1) = -2 \quad T(4) = 7$ linear

$T(2) = 1 \quad T(5) = 10 \quad O(n)$

$T(3) = 4 \quad T(6) = 13$

$T(n) \leq c \cdot n \quad c \geq 0$ desde que $n > k$

$cn \leq 3c(n-2)+1 \leq cn$

$cn - 2c + 1 < cn$

$cn \leq 3cn - 6c + 1 \leq cn$

$n = 1$

$c \leq 3c - 6c + 1 \leq c$

$-3c + 1 \leq c$

$+4c \geq 1$

$\boxed{c \geq \frac{1}{4}}$

$c \leq 3c - 6c + 1$

$4c \leq 1$

$\boxed{c \leq 1/4}$

$\boxed{c = 1/4} \rightarrow$ p/ ser $\Theta(n)$