Data Structures Quick Test

Data Structures

Q1: Which of the following data types allows the access to the first and last element in constant time $O(1)$?

○ Arrays ○ Lists with first pointer only ○ Lists with pointer first and last pointer
(Note that this question contained *removal*, instead *access to* in the previous version)

Q2: What is the runtime complexity to access the nth element of a list?

○ $O(1)$ ○ $O(\log(n))$ ○ $O(n)$

Q3: What is the runtime complexity to access the nth element of an array?

○ $O(1)$ ○ $O(\log(n))$ ○ $O(n)$

Q4: If $f(n) \in O(g(n))$ holds then $g(n) \in O(f(n))$ also holds.

○ True ○ False

Q5: When implementing a stack using a singly-linked list, then the order of elements in the list has no impact on the runtime behaviour of the implementation.

○ True ○ False

Q6: Stacks can be implemented using linked lists and arrays, whereas queues can only be implemented with linked lists, but not arrays.

○ True ○ False

Q7: $n^2 \in O(n)$.

○ True ○ False

Q8: Given a runtime function $T(n) = 2 * n^2 + 4 * \log(n) + 5$, which of the following statements is correct?

○ $T(n) \in O(n)$ ○ $T(n) \in O(n \log(n))$ ○ $T(n) \in O(n^2)$ ○ $T(n) \in O(n^2 \log(n))$

Q9: What is the amortized runtime behaviour of n push_back operations if we always increase the vector size by a constant c when we resize the array?

○ $O(1)$ ○ $O(\log(n))$ ○ $O(n)$

Q10: What is the amortized runtime behaviour of n push_back operations if we always double the size of the vector every time we resize the array?

○ $O(1)$ ○ $O(\log(n))$ ○ $O(n)$