Quizizz

NAME : ______________________

CLASS : ______________________

DATE : ______________________

DS UNIT-2 TEST-2

13 Questions

1. What are the time complexities of finding 8th element from beginning and 8th element from end in a singly linked list? Let n be the number of nodes in linked list, you may assume that n > 8.

A O(1) and O(1)

B O(n) and O(n)

C O(n) and O(1)

D O(1) and O(n)

2. Is it possible to create a doubly linked list using only one pointer with every node.

A Yes, possible by storing XOR of current node and next node

B Yes, possible by storing XOR of current node and previous node

C Not Possible

D Yes, possible by storing XOR of addresses of previous and next nodes

3. Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?

A Possible if X is not first node. Use following two steps (a) Copy the data of next of X to X. (b) Delete next of X.

B Possible if size of linked list is odd

C Possible if X is not last node. Use following two steps (a) Copy the data of next of X to X. (b)Update the pointer of node X to the node after the next node. Delete next of X.

D Possible if size of linked list is even

4. Which of the following is an application of XOR-linked lists?

A Implementing stacks

B Caching data structures

C Implementing queues

D Memory-efficient linked list representation

5. Consider the following function to traverse a linked list.

```
void traverse(struct Node *head)
{
  while (head->next != NULL)
  {
     printf("%d  ", head->data);
     head = head->next;
  }
}
```

Which of the following is **FALSE** about above function?

A The function is implemented incorrectly because it changes head

B The function doesn't print the last node when the linked list is not empty

C None of the Above

D The function may crash when the linked list is empty

6. Let P be a singly linked list. Let Q be the pointer to an intermediate node x in the list. What is the worst-case time complexity of the best known algorithm to delete the node Q from the list?

A O(1)

B O(log2 n)

C O(n)

D O(log n)

7. N items are stored in a sorted doubly linked list. For a delete operation, a pointer is provided to the record to be deleted. For a decrease-key operation, a pointer is provided to the record on which the operation is to be performed. An algorithm performs the following operations on the list in this order: Θ(N) delete, O(log N) insert, O(log N) find, and Θ(N) decrease-key What is the time complexity of all these operations put together?

A O(N Log N)

B O(N)

C $O(\text{Log}^2 N)$

D $\Theta(N^2 \text{ Log } N)$

8. The concatenation of two lists is to be performed in O(1) time. Which of the following implementations of a list should be used?

A circular doubly linked list

B doubly linked list

C singly linked list

D array implementation of lists

9. Consider the following piece of 'C' code fragment that removes duplicates from an ordered list of integers.

```
Node remove-duplicates(Node head, int j) {
Node t1, t2; j=0; t1 = head;
if (t1! = NULL) t2 = t1 →next;
else return head;
j = 1;
if(t2 == NULL)
return head;
while t2 != NULL) {
if (t1.val != t2.val) --------------------------→ (S1) {
(j)++; t1 -> next = t2; t1 = t2: ----------→ (S2) }
t2 = t2 →next; }
t1 →next = NULL;
return head; }
```

Assume the list contains *n* elements (n≥2) in the following questions. a). How many times is the comparison in statement *S1* made? b). What is the minimum and the maximum number of times statements marked *S2* get executed? c). What is the significance of the value in the integer pointed to by *j* when the function completes?

A

(a). n-1 times, since comparison is pairwise for n elements.
(b). maximum : n-1 for all distinct elements, minimum: 1 for all same elements.
(C). j keeps count of distinct nodes in the list.

B

(a). n times, since comparison is pairwise for n elements.
(b). maximum : n-1 for all distinct elements, minimum: 0 for all same elements.
(C). j keeps count of distinct nodes in the list.

C

(a). n-1 times, since comparison is pairwise for n elements.
(b). maximum : n-1 for all distinct elements, minimum: 0 for all same elements.

D

None of the Above

(C). j keeps count of distinct nodes in the list.

10. Suppose there are two singly linked lists both of which intersect at some point and become a single linked list. The head or start pointers of both the lists are known, but the intersecting node and lengths of lists are not known. What is worst case time complexity of optimal algorithm to find intersecting node from two intersecting linked lists?

A Θ(m+n), where m, n are lengths of given lists

B Θ(n^2), where m>n and m, n are lengths of given lists

C Θ(min(n, m)), where m, n are lengths of given lists

D Θ(n*m), where m, n are lengths of given lists

11. S1 : Anyone of the followings can be used to declare a node for a singly linked list.
If we use the first declaration, "struct node * nodePtr;" would be used to declare pointer to a node.
If we use the second declaration, "NODEPTR nodePtr;" can be used to declare pointer to a node.
/* First declaration / *struct node { int data; struct node* nextPtr; };
/* Second declaration / *typedef struct node{ int data; NODEPTR nextPtr; }* NODEPTR;

A Statement S1 is FALSE

B Statement S1 is TRUE

12. In a doubly linked list, the number of pointers affected for an insertion operation will be

A can not say

B 1

C 5

D 0

13. Consider an implementation of unsorted single linked list. Suppose it has its representation with a head and a tail pointer (i.e. pointers to the first and last nodes of the linked list). Given the representation, which of the following operation can not be implemented in O(1) time ?

A Deletion of the front node of the linked list

B Deletion of the last node of the linked list.

C Insertion at the end of the linked list

D Insertion at the front of the linked list

Answer Key

1. d
2. d
3. c
4. d
5. a
6. c
7. a
8. a
9. c
10. a
11. a
12. a
13. b