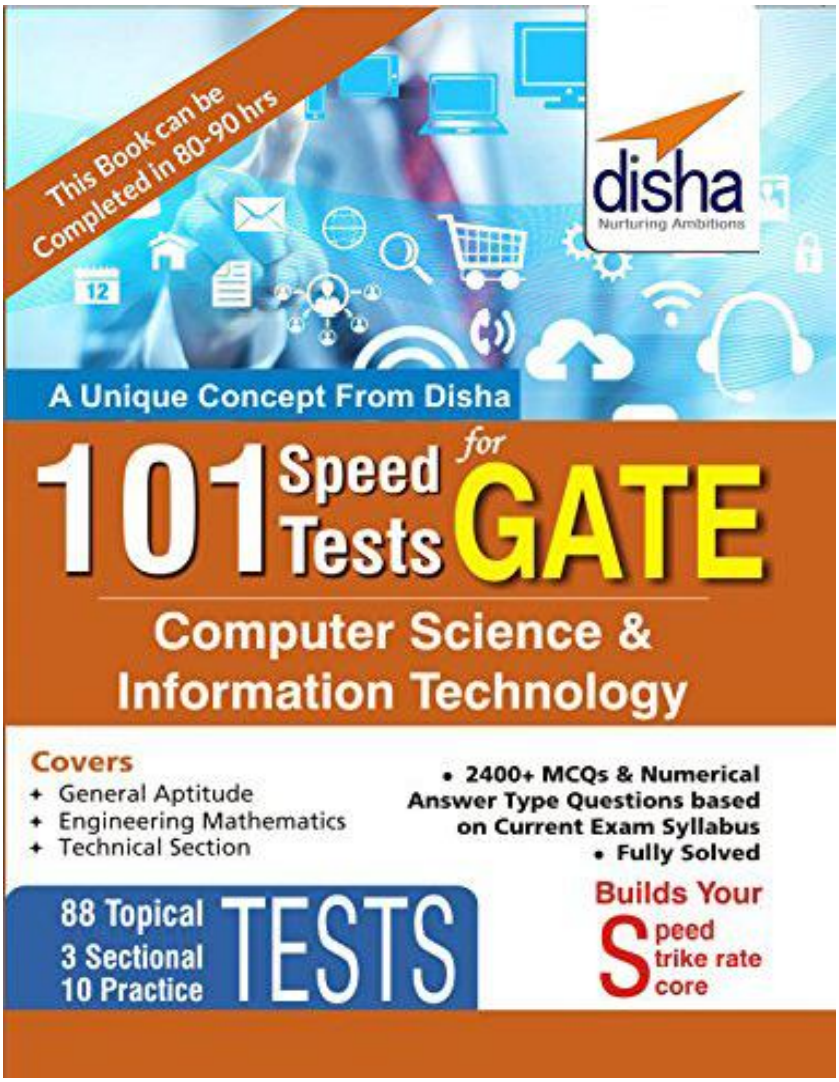
Linked Lists

This Chapter is taken from our Book:



ISBN : 9788193288979

101 SPEED TEST

43

Linked Lists

Max. Marks : 22 | No. of Qs. 22 | Time : 30 min. | Date : ...../...../.....

1. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
        int value;
        struct node *next;
};
void rearrange (struct node *list){
        struct node *p, *q;
        int temp;
        if (!list || !list -> next) return;
        p = list, q = list –> next;
        while (q){
        temp = p - > value; p -> value = q - > value;
        q - > value = temp; p = q - > next;
        q = p?p -> next: 0;
        }
        }
```

(a) 1, 2, 3, 4, 5, 6, 7 (b) 2, 1, 4, 3, 6, 5, 7
(c) 1, 3, 2, 5, 4, 7, 6 (d) 2, 3, 4, 5, 6, 7, 1

2. The following C function takes a simply-linked list as input argument. It modifies the list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
type def struct node {
        int value;
        struct node *next;
}       Node*;
Node *move_to_front (Node *head) {
        Node *p, *q;
if (head = = NULL || (head -> next = = NULL)) return head;
        q = NULL; p = head;
        while (p - > next ! = NULL)     {
        q = p;
        p = p - > next;
}
return head;
}
```

Choose the correct alternative to replace the blank line.

(a) q = NULL; p - > next = head; head = p;
(b) q - > next = NULL; head = p; p - > next = head;
(c) head = p; p -> next = q; q -> next = NULL;
(d) q-> next = NULL; p-> next = head; head = p;

3. When a new element is inserted in the middle of a linked list, then
(a) only elements that appear after the new element need to be moved
(b) only elements that appear before the new element need to be moved
(c) elements that appear before and after the new element need to be moved
(d) None of these

4. If address of the 8th element in a linked list of integers is 1022, then address of 9th elements is
(a) 1024 (b) 1026
(c) 1023 (d) None of these

5. Which of the following operations is performed more efficiently by doubly linked list than by linear linked list?
(a) Deleting a node whose location is given
(b) Searching an unsorted list for a given item
(c) Inserting a node after the node with a given location
(d) Traversing the list to process each node.

6. Consider a linked list implementation of a queue with two pointers: front and rear. The time needed to insert element in a queue of length n is
(a) O(1) (b) O ($\log_2 n$)
(c) O(n) (d) O($n \log_2 n$)

7. The linked list implementaion of sparse matrices is superior to the generalized dope vector method because it is
(a) conceptually easier and completely dynamic
(b) efficient if the sparse matrix is a band matrix
(c) efficient in accessing an entry
(d) all of these

8. In a circularly linked list organization, insertion of a record involves the modification of
(a) no pointer (b) 1 pointer
(c) 2 pointers (d) 3 pointers

9. Which of the following lines of code will delete two successive nodes of a singly linked linear list (WITH MORE THAN 2 NODES)?
Assume this code is in the main program, not a subproceduce?
(a) LINK [X] : = LINK [LINK] [X]];
(b) X : = LINK [LINK [X]];
(c) LINK [LINK [X]]: = X;
(d) LINK [X] : = LINK [LINK [LINK [X]]]:

10. To free which of the following list traversing through the entire list is not necessary?
(a) Circular list (b) Singly linked list
(c) Double linked list (d) Both (b) and (c)

11. Which of the following statement(s) is/are true regarding insertion of node in a linear linked list?
(a) Setting the field of the new node means allocating memory to newly created node
(b) If node precedes all other is the list, then insert it at the front and return its address
(c) Creating a new node depends upon free memory space
(d) All of these

12. Identity the steps to be taken when a first node is to be deleted from linear linked list.
I. Set link of start pointer to the second node in the list
II. Free the space associated with first node
III. Obtain the address of the second node in the list
IV. Count the number of nodes in the list.
Codes:
(a) I and II (b) I, II and III
(c) II and III (d) I, II, III and IV

13. 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) Singly linked list
(b) Double linked list
(c) Circular doubly linked list
(d) Array implementation of list

14. A linked binary tree with n nodes. $n \geq o$ has exactly
(a) 2 n + 1 NULL links (b) 2 n–1 NULL links
(c) n + 1 NUL links (d) n – 1 NULL links

15. The time required to search an element in a linked list of length n is
(a) O (n) (b) O (1)
(c) O (n^2) (d) O ($\log_2 n$)

16. the minimum number of fields with each node of double linked list is .....?

17. If we want to find last node of a linked list then the correct coding is
(a) If (temp → link! = NULL)
temp = temp → link
(b) If (temp → data = Num)
temp = temp → link
(c) while (temp → link! = NULL)
temp = temp → link
(d) while (temp → link! = data)
temp = temp → link

18. Consider the following function that takes reference to head of a Doubly Linked List as parameter. Assume that a node of doubly linked list has previous pointer as prev and next pointer as next.

```
void fun(struct node **head_ref)
{
  struct node *temp = NULL;
  struct node *current = *head_ref;
  while (current !=  NULL)
  {
    temp = current->prev;
    current->prev = current->next;
    current->next = temp;
    current = current->prev;
  }

  if(temp != NULL )
    *head_ref = temp->prev;
}
```

Assume that reference of head of following doubly linked list is passed to above function 1↔ 2 ↔ 3 ↔ 4 ↔ 5 ↔ 6. What should be the modified linked list after the function call?
(a) 2 ↔ 1 ↔ 4 ↔ 3 ↔ 6 ↔ 5
(b) 5 ↔ 4 ↔ 3 ↔ 2 ↔ 1 ↔ 6
(c) 6 ↔ 5 ↔ 4 ↔ 3 ↔ 2 ↔ 1
(d) 6 ↔ 5 ↔ 4 ↔ 3 ↔ 1 ↔ 2

19. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```
/* Link list node */
struct  node
{
  int data;
  struct node* next;
};

/* head_ref is a double pointer which points to head (or start) pointer
of linked list */
static void reverse(struct node** head_ref)
{
  struct node* prev  = NULL;
  struct node* current = *head_ref;
  struct node* next;
  while (current != NULL)
  {
    next  = current->next;
    current->next = prev;
    prev = current;
    current = next;
  }
  /*ADD A STATEMENT HERE*/
}
```

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list.
(a) *head_ref = prev; (b) *head_ref = current;
(c) *head_ref = next; (d) *head_ref = NULL;

20. 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: ?(N) delete, O(log N) insert, O(log N) find, and ?(N) decrease-key What is the time complexity of all these operations put together
(a) O(Log^2N) (b) ?(N^2 Log N)
(c) O(N) (d) O(N^2)

21. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct  node
{
 int value;
 struct node *next;
};
void rearrange(struct node *list)
{
 struct node *p, * q;
 int temp;
 if ((!list) || !list->next)
    return;
 p = list;
 q = list->next;
 while(q)
 {
   temp = p->value;
   p->value = q->value;
   q->value = temp;
   p = q->next;
   q = p?p->next:0;
 }
}
```

(a) 1,2,3,4,5,6,7 (b) 2,1,4,3,6,5,7
(c) 1,3,2,5,4,7,6 (d) 2,3,4,5,6,7,1

22. 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(n) (b) O(1) and O(1)
(c) O(n) and O(1) (d) O(n) and O(n)

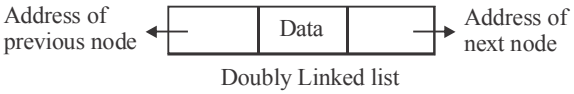
Hints & Solutions

1. **(b)** The function rearrange() exchanges data of every node with its next node. It starts exchanging data from the first node itself.

2. **(d)** When the while loop ends, q contains address of second last node and p contains address of last node. So we need to do following things after while loop.
(i) Set next of q as NULL (q->next = NULL).
(ii) Set next of p as head (p->next = head).
(iii) Make head as p (head = p)
Step (ii) must be performed before step (iii). If we change head first, then we lose track of head node in the original linked list.

3. **(d)** **4.** **(d)** **5.** **(a)** **6.** **(a)** **7.** **(d)**
8. **(c)** **9.** **(d)** **10.** **(d)** **11.** **(d)** **12.** **(a)**
13. **(c)** **14.** **(c)** **15.** **(a)**

16. **3**



Doubly Linked list

17. **(c)**

18. **(c)**

19. **(a)** *head_ref = prev; At the end of while loop, the prev pointer points to the last node of original linked list. We need to change *head_ref so that the head pointer now starts pointing to the last node.

20. **(d)**

21. **(b)** The function re-arrange () exchanges data of every node with its next node. It starts exchanging data from the first node itself.

22. **(a)** Finding 8th element from beginning requires 8 nodes to be traversed which takes constant time. Finding 8th from end requires the complete list to be traversed.