5
CHAPTER

Linked Lists

Q.1 The concatenation of two lists is performed in $O(1)$ time. Which of the following implementations of a list should be used?
(a) singly linked list
(b) doubly linked list
(c) circular doubly linked list
(d) array implementation of list

Q.2 Which data structure would be most appropriate to implement a collection of values with the following three characteristics?
- Items are retrieved and removed from the collection in FIFO order.
- There is no prior limit on the number of items in the collection.
- The size of an item is large relative to the storage required for a memory address.

(a) Singly-linked list, with head and tail pointers
(b) Doubly-linked list, with only a head pointer
(c) Binary tree
(d) Hash table

Q.3 Assume given a non-empty binary search tree. If its root is passed to the following function then find the modified binary search tree?
```
void find (struct node *head)
{
    struct node * x;
    x = head;
    while (x → left ! = NULL)
    {
        x = x → left;
        if (x → left = = NULL);
        {head → data = x → data;}
    }
}
```
(a) It modifies the root data with the left node data
(b) It modifies the root data with the random left node data
(c) It modifies the root data with minimum value of tree.
(d) It modifies the root data with maximum value of tree.

Q.4 Consider the following code which is used to detect the loop in the linked list. Find the missing statement A?
```
p = head;
q = head → next;
while (A)
{
    if(p == q) exit (0); //loop detected
    p = p → next;
    q = (q → next)? (q→next → next): q → next;
}
```
(a) p! = NULL
(b) q! = NULL
(c) (p! = NULL) && (q ! = NULL)
(d) (p! = NULL)|| (q ! = NULL)

Q.5 Consider the following code
```
Node *find (Node * head)
{
   Node *P1 = head, *P2 = head;
   while (P2)
   {
      P1 = P1 → next;
      P2=(P2→next)? P2 → next → next: NULL;
   }
   printf ("%d", P1 → value);
}
```
Assume Node is the structure type with two members: 'value' and 'next'. Identify the node value printed by the above code if non-empty linked list header is passed to the function 'find'.

(a) First element of the list [i.e., value of the first node]
(b) Second element of the list
(c) Middle element of the list
(d) Last element of the list

Q.6 The function **delete (head, element)** is used to delete a node from the linked list by finding the node value with a given element. The parameter head is the first node of the list. Find the missing statements *A* and *B* in the following "delete" function to delete the node? (Assume all elements are distinct in the list and the function returns pointers that points to the first node of the list).

```
Node delete (Node head, int element)
{
    Node x = head;
    if (x.data == element)    return head.next;
    while (x.next ! = NULL)
    {
        if    (_____A_____)
        {
            _____B_____;
            return head;
        }
        x = x.next;
    }
}
```

(a) *A* : *x*.data == element
B : *x*.next = *x*.next.next
(b) *A* : *x*.next.data == element
B : *x*.next = *x*.next.next
(c) *A* : *x*.data == element
B : *x*.next.next = *x*.next
(d) *A* : *x*.next.data == element
B : *x*.next.next = *x*.next

Q.7 Identify the functionality of the following code when function returns the value 1. ["head" pointer point to the first node of the non-empty linked list]

```
int find (Node *head)
{
    Node *P1 = head, *P2 = head;
    if (head → next != NULL)
    {
        P1 = head → next;
        P2 = (head → next)? head →next → next:
        NULL;
    }
    while ((P1! = NULL) && (P2! = NULL))
    {
        if (P1 == P2)  return 1;
        P1 = P1 → next;
        P2 = (P2 → next != NULL)? P2 → next →
        next: NULL;
    }
    return 0;
}
```

(a) It finds the duplicate element in the list
(b) It finds the middle node in the list
(c) It finds the cycle in the list
(d) It finds the last node in the list

Q.8 Let 'new' and 'current' be two pointers. 'new' is pointing to the new node and 'current' is pointing to the some node in the doubly linked list. If new node needs to be inserted after the current node then find which of the following operations are correct?

(a) new → next = current → next;
new = current → next;
new → prev = current;
(current → next) → prev = new;
(b) new → next = current → next;
new → next = current;
new → prev = current;
(new → next) → prev = new;
(c) new → next = current → next;
current → next = new;
new → prev = current;
(new → next) → prev = new;
(d) new → next = current → next;
current → next = new;
new → prev = current;
(current → next) → prev = new;

Q.9 Struct void find (Struct node ** head)

```
{   Struct node*X = *head;
    Struct node* Y;
    Struct node * Z = NULL;
    while (X != NULL)
    {
        Y = X → next ;
        X → next = Z ;
        Z = X ;
        X = Y ;
    }
    *head = Z ;
}
```

If head is a pointer to a pointer to the first node of the list and it is passed to the 'find' function then find the list after executing the function.

(a) It adds a new node at the first
(b) It adds a new node at the last
(c) It keeps the list as same
(d) It reverses the list

Q.10 Consider the following structure for creating nodes of a double linked list.

```
struct node
{
    int data;
    struct node * x1 ; /* left pointer */
    struct node * y1 ; /* right pointer */
};
```

Let us assume P be the node in the existing linked list. Which of the following is true about the following code C_1?

Code C_1 : $\boxed{\begin{array}{l}(P \to x_1) \to y_1 = P \to y_1; \\ (P \to y_1) \to x_1 = P \to x_1;\end{array}}$

Assume memory is made free automatically by the compiler after deletion.

(a) It deletes the node to the right of P
(b) It deletes the node to the left of P
(c) It deletes the node P from the linked list
(d) None of the above

Q.11 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 size of linked list is even
(b) Possible if size of linked list is odd
(c) Possible if X is not first node
(d) Possible if X is not last node: (a) copy the data of next of X to X (b) delete next of X.

Q.12 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

Q.13 The following C function takes single linked list of integers as parameter and rearrange the elements of the list. The function is called with the list containing integers 10, 20, 30, 40, 50, 60, 70 in given order and generate output as 20, 10, 40, 30, 60, 50, 70 after completion. What will be the correct options files so that we get desired output?

```
struct node
{
    int val;
    struct node * next;
}
void Altswap(struct node * list)
{
    struct node *p, *q;
    int temp;
    if (! list || list → next) return;
    p = list; q = list → next;
    while (q)
    {
        W
        X
        Y
        Z
        A
    }
}
```

(a) $W : p \rightarrow$ val $= q -$ val;
$X : q \rightarrow$ val $=$ temp;
$Y : p = q \rightarrow$ next;
$Z :$ temp $= p \rightarrow$ val;
$A : q = p \,?\, p \rightarrow$ next $: 0$;

(b) $W :$ temp $= p \rightarrow$ val;
$X : q \rightarrow$ val $=$ temp;
$Y : p \rightarrow$ val $= q \rightarrow$ val;
$Z : q = p \,?\, p \rightarrow$ next$: 0$;
$A : p = q \rightarrow$ next;

(c) $W : q = p \,?\, p \rightarrow$ next$: 0$;
$X :$ temp $= p \rightarrow$ val;
$Y : p \rightarrow$ val $= q \rightarrow$ val;
$Z : q \rightarrow$ val $=$ temp;
$A : p = q \rightarrow$ next;

(d) $W :$ temp $= p \rightarrow$ val;
$X : p \rightarrow$ val $= q \rightarrow$ val;
$Y : q \rightarrow$ val $=$ temp;
$Z : p = q \rightarrow$ next;
$A : q = p \,?\, p \rightarrow$ next $: 0$;

■■■■

Linked List

DATE

PAGE

(1) concatenation → O(1)

(d) → Circular doubly linked list Ans

→ ~~Binary Tree Ans~~

(2) ⊙ retrive & remove → FIFO

(a) ⊙ no limit collection

⊙ size item large

→ ~~Binary Tree~~ → ~~Ans~~

(c) → Singly linked list, with head & tail pointer Ans

(3) (c)

```
Void find ( struct node *head)
{
  struct node *x;
   x = head;
    while (x → left != NULL)
    {   x = x → left;
         if (x → left == NULL);
         { head → data = x → data; }
    }
}
```

→ It modifies the root data with minimum value of tree

Ans

4)→

(c)

```
p = head;
q = head->next;
while ( A)
{
    if (p == q)   exit(0);
    p = p -> next;
    q = (q -> next) ? (q -> next -> next) : q -> next;
}
```

→ (p != NULL) && q != NULL)   Ans

5)→

(c)

```
Node *find (Node *head)
{ Node *P1 = head, *P2 = head;
  ~~{ Node *P1 = head, *P2 =~~
  while (P2)
  { P1 = P1 -> next;
    P2 = (P2 -> next)? P2 -> next -> next : NULL;
  }
  printf("%d", P1 -> value);
}
```

→ middle element of list

Ans

6 ⇒ B

```
Node delete ( Node head, int element )
{    Node  x = head;
     if ( x. data == element )
             return  head.next;
     while ( x . next != NULL)
     {  if ( ____A____ )
         {
               ____B____ ;
               return  head;
         }
         x  = x. next;
     }
}
```

A : x.next.data == element
B : x . next = x.next.next

Ans

7 ⇒ C

find cycle in list

Ans

```
int find ( Node * head)
{    Node * P1 = head ;  * P2 = head ;
     if (head -> next != NULL)
     { P1 = head -> next ;
       P2 = ( head -> next )? head -> next -> next : NULL;
     }
     while ((P1 != NULL) && (P2 != NULL))
     { if (P1 == P2) return 1;
       P1 = P1 -> next;
       P2 = (P2 -> next != NULL) ? P2 -> next -> next : NULL; }
   return 0;
}
```

(8) Insert after current node = ?

~~Next~~

(6)
```
new ->next = current ->next;
current ->next = new;
new->prev = current;
(new->next) -> prev = new;
```

Ans

(9)
```
struct   void find (struct node **head)
{ struct node *x = *head;
  struct node *y;
  struct node *z = NULL;
  while (x != NULL)
  {  y = x ->next;
     x -> next = z;
     z = x;
     x = y;
  }
  *head = z;
}
```

(D)

reverse the list

Ans

(10) $(p \to x_1) \to y_1 = p \to y_1;$
$(p \to y_1) \to x_1 = p \to x_1;$

(c)

Memory free after deletion —

→ It deletes the node P from linked list

Ans

(11) point X in singly linked list,
x pointing ~~node~~ not given
(Can delete X = ?

→ Possible, if X is not last node,
⊙ Copy the data of next of X of x.
⊙ delete next of X.

Ans

(12) circularly linked list, insertion of record involve in modification,

(a)

→ 2 pointers Ans

(137)

(D)

```
void AIsorp ( struct node *list )
{
    struct node *p, *q;
    int temp;
    if(! list || list->next)
                return;
     p = list;
     q = list->next;
      while (q)
    {
        W
        X
        Y

        Z
        A
    }
}
```

```
W : temp = p->val;
X : p->val = q->val;
Y : q->val = temp;
Z : p = q->next;
A : q = p ? p->next : 0;
```

Ans