Page No.
Date

Practice Quiz
Linklist

(1) p = p→next;

(2) p = q;

(3) q = p;

4 r = (p→next);

5 p→data = r→data;

6
```
Node *r          //(pointer of Node type).
  r = p;         //(Address of first node).
do {                  //(Traversing Linklist)
          r = r→next;
  } while (r→next != NULL)
  p→data = r→data;
```

7 r→next = p;

8 (q→next)→next = p;

9
```
Node *end //(pointer of Node type)(to find last node)
  end = start //(Address of starting node)
do{                 //(Traversing Linked List)
      end = end→next
  } while (end→next != null)
  end→next = start;
```

Page No.
Date

```
10  p → next = q;

11  Node *p, *q                    // Pointer of Node type
    p = start                      // Point to starting node
    q = start → next               // Point to node after start
     while (   q → next != NULL)
     {                                      // Traverse List
         if ( q → data == 'B')  // Check for Condition
         {
             p → next = q → next; // If True than
             break;                 update address
         }
         else
         {                                  // Keep searching
             p = p → next
             q   = q → next
         }
     }
     free (p);
     free (q);                     // Free memory of Node

12  Node  * q                      // Traversing of Linked List
    q = start;
    while ( q → next != NULL)
    {
         q = q → next;
    }
```

Page No.
Date

```
13 Node *q                    // Pointer of Type node
   q = start;                 // Searching Algorithm
   while(q -> data != 'C')    // check for
   {                             Condition
       q = q -> next          // Advance pointer
   }

14 Node *p, *q;   // Pointer of Type node.
   int i;         // counter for for loop
   for (i=0; i< 4;i++)
   {
                                  // Create new node
      p = (Node*)malloc(sizeof(Node));
      p -> next = NULL;   // set address of next node to
                                                     null.
        ~~scanf("%c",&(p->data));~~
        if(Head = NULL)                   // A for 0, B for 2
      {                                      C for 3, D for 4
        head = q = p;     // If first Node
      }
      else
       {
          q -> next = p;   // If ~~Next~~ not first
          q = q -> next;      Node.
       }
   }
```

```
15  Node   *q;                                   //Pointer of type Node
    q = (Node *) malloc (sizeof (Node));         //Create Node
    q ->data = 'A';                              //store data.
                                                   address
    q ->next =  p;                               //Update ~~data~~ of New Node
    p = q;                                       //set New node as first

16  Node        *q, *r;    //Pointers of type Node.

    r = p;               //    Point r to starting Part
     q = (Node*) malloc(sizeof(Node)); //Create Node
     q->data = 'D';                    // Store Data of Node
     q->next = NULL;                   // Store address as Null
    while ( r->next != NULL)
    {                                  // Traverse LL To find
          r = r->next;                 Last Node.
    }
     r->next = q;                      //set New Node as last Node

17   q = p;                 //set pointer to  Node A.

    while( q->next != NULL)
    {                          // Traverse LL to go till Node C
          q = q->next;
     }
     q->next = p;     // Set next node address to Node A
     p = p->next;     //Set starting Node to be Node B
     q = q->next;     //set pointer to Node A.
                               next
     q->next = NULL;  // Set address of A to Null
```

Page No.
Date

```
18  q = p.                    // set pointer to node A.
    r = p -> next ;          // set pointer to node B
    p = p -> next;           // set start to node B
    q -> next = NULL ;       // set next address to null

    while( p != NULL)
     {
        p = p -> next;       // Set start to next node
        r -> next = q ;      // set address of next node to
        q = r;             } previous
        r = p;             } // update values
     }
       p = q ;               // Set starting address to Node D

19  Node *s                  // Declare pointer of type Node
    if ( p -> data < q -> data ) // Check for
    {                              Condition
        r = s = p;           // First Node
        p = p -> next;       // update p
    }
    else
    {
        r = s = q;
        q = q -> next;       // update q
    }
    while (p -> next != NULL && q -> next != NULL) // while
    {                                                both are not empty
      if (p -> data < q -> data)
      {
          s -> next = p;     // Assign next address
          p = p -> next;     // update p
          s = s -> next      // update s
      }
```

```
else
{
    s -> next = q;
    q = q -> next; s = s -> next;
}
}
while (p -> next != NULL) //If p is not empty
{
    s -> next = p;
    p = p -> next; s = s -> next;
}
while (q -> next != NULL) //If q is not empty
{
    s -> next = q;
    q = q -> next; s = s -> next;
}
```