CSN-102 Data Structures
Topic : Linked Lists
Tutorial 2

—---

Tutorial Questions

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 by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
typedef 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;
}
```

a) q = NULL; p→next = head; head = p;
b) q→next = NULL; head = p; p→next = head;
c) p→next = head; head = p; q→next = NULL;
d) q→next = NULL; p→next = head; head = p;

3) 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) $O(n^2)$
b) O(1)
c) O(n)
d) O(log n)

4) Consider the function f defined below.

```
struct item
{
    int data;
    struct item * next;
};

int f(struct item *p)
{
    return ((p == NULL) || (p->next == NULL)   ||
    ((P->data <= p->next->data) && f(p->next)));
 }
```

What does this function ensure ?

a) the list is empty or has exactly one element
b) the elements in the list are sorted in non-decreasing order of data value
c) the elements in the list are sorted in non-increasing order of data value
d) not all elements in the list have the same data value

5) In a circular linked list organization, insertion of a record at end involves modification of

a) One pointer
b) Two pointers
c) Multiple pointers

d) No pointer

6) What is the output of the function when p points to linked list 9, 4, 5, 2, 7.

```
 void funct(struct node *p)
  {
       if (p)
       {
          funct (p->link) ;
          printf (“%d “, p->data) ;
       }
  }
```

7) 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?

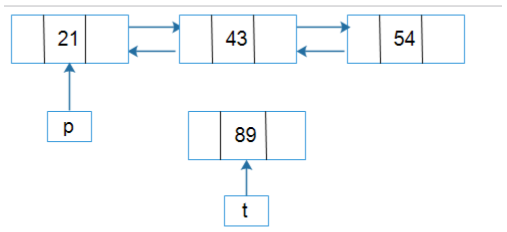
8) Please fill the lines in the code segment to reverse a single linked list.

```
struct item
{
   int data;
   struct item * next;
};

struct node *rev (struct node *cur)
{
   struct *prev =NULL, *nextNode = prev;
   while (cur)
   {
         ?

   }
   return  —---- ;
}
```

9) Write the code segment to insert the new node t in the middle of the doubly linked list shown below.

```
struct node
```

```
{
   int x ;
   struct node *prev ;
   struct node *next ;
} ;
```


21
43
54
p
89
t