Due on 15th Sep 2020 / A5

Week 5- Lsit ADT, singly linked list

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(n)

B O(1) and O(1)

C O(n) and O(1)

D O(n) and O(n)

Ans: (a) O(1) and O(n)

for finding the 8th element from the beginning , we can simply go from first to 8th element and locate it .There is no need of finding n or number of elements

Whereas for finding the 8th element from the end,we need to first find the last element and then go 8 steps backwards . So time depends on n value.

2. You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?

A Delete the first element

B Insert a new element as a first element

C Delete the last element of the list

D Add a new element at the end of the list

Ans: (C) Delete the last element of the list

Once we delete the last node , we will need to update the node before this deleted node as the last node.For that we will need to traverse from beginning of the list to the n-1th node.So it is dependent on the length.

3. 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 may crash when the linked list is empty

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

C The function is implemented incorrectly because it changes head

Ans: (c) The function is implemented incorrectly because it changes head.
this statement is false as changing head won't bring any difference in this case as we only want to traverse from start to end.

4. 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)

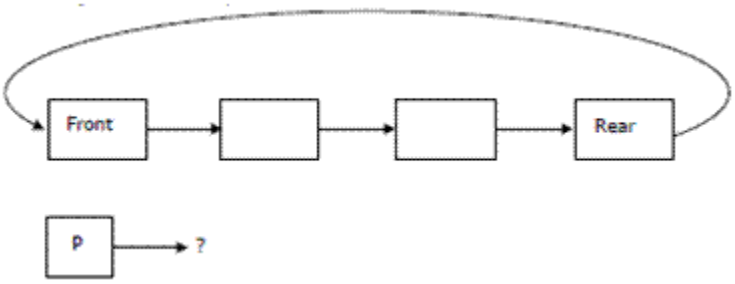
B O(1) and O(1)

C O(n) and O(1)

D O(n) and O(n)

Ans:(A) O(1) and O(n)
for finding the 8th element from the beginning , we can simply go from first to 8th element and locate it .There is no need of finding n or number of elements
Whereas for finding the 8th element from the end,we need to first find the last element and then go 8 steps backwards . So time depends on n value.

5. A circularly linked list is used to represent a Queue. A single variable p is used to access the Queue. To which node should p point such that both the operations enQueue and deQueue can be performed in constant time?



A rear node

B front node

C not possible with a single pointer

D node next to front

Ans: (a) rear node
If p points rear node then if we want to enqueue something,we can directly enqueue at the rear if we want to dequeue also ,we can go to the next node of rear (front) and dequeue from there.

6. What is the output of following function for start pointing to first node of following linked list? 1->2->3->4->5->6

```
void fun(struct node* start)
{
  if(start == NULL)
    return;
  printf("%d  ", start->data);

  if(start->next != NULL )
    fun(start->next->next);
  printf("%d  ", start->data);
}
```

A 146641

B 135135

C 1235

D 135531

Ans: (D) 1 3 5 5 3 1

7.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 Error, No output

Ans: (B) 2 1 4 3 6 5 7

8. 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))
         );
}
```

For a given linked list p, the function f returns 1 if and only if

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.

Ans: (b)

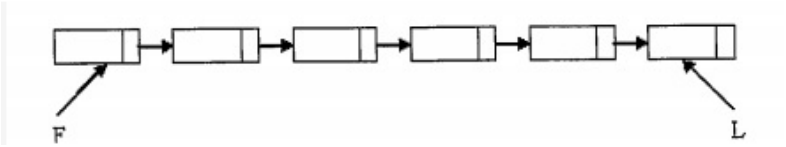
The function returns 1 for the below 3 cases:

1) when the list is empty
2) when the list has only one node
3) when the elements in the list are in increasing order

This is in short expressed as Option B

9.

Consider a single linked list where F and L are pointers to the first and last elements respectively of the linked list. The time for performing which of the given operations depends on the length of the linked list?



a) Delete the first element of the list
b) Interchnage the first two elements of the list
c) Delete the last element of the list
d) Add an element at the end of the list

Ans:

(c) Delete the last element of the list

Once we delete the last node , we will need to update the node before this deleted node as the last node.For that we will need to traverse from beginning of the list to the n-1th node.So it is dependent on the length