{{{questionNumber}}}. Consider a class `List` that is implemented using a doubly linked list with only a `head` pointer (i.e. pointer to the first node in the list).

Given that implementation, which of the following operations *cannot* be implemented in O(1) time?

I. Insert item at the front of the list
II. Insert item at the rear of the list
III. Delete front item from list
IV. Delete rear item from list

A. I, II and III
B. All of them
C. [Your Answer] I and III
D. I and II
E. [Correct Answer] II and IV

{{{questionNumber}}}. In a sorted doubly linked list containing n nodes, the time taken to print out the 1st, 2nd, 4th, 8th, 16th, etc. elements is:

[Correct Answer] A. O(n).
B. O(log n).
[Your Answer] C. O(1).
D. $O(n^2)$.
E. O(n log n).

{{{questionNumber}}}. Which of the following List ADT implementations gives us an O(1) time for `insertAtFront`, i.e inserting an element at the front of the list?

I. A singly-linked list with only a `head` pointer.
II. A singly-linked list with `head` and `tail` pointers.
III. A doubly-linked list with only a `tail` pointer.
IV. A doubly-linked list with `head` and `tail` pointers.

A. II and IV
B. I, II, III and IV
C. I and II
D. None of the other options is correct
E. [Correct Answer] [Your Answer] I, II and IV

{{{questionNumber}}}. Consider the following function definition and suppose that 1) the `node` class consists of an integer `data` element, and a `node` pointer called `next`, and 2) variable `head` is the address of a linked list of such `node`s.

What does the function do?

```
void fun(node *
curr) {    if
(curr != NULL)
{       cout <<
curr->data;
fun(curr->next);
   } }

node * head = NULL;
// maybe insert data into the chain
here fun(head);
```

A. `fun` prints the reverse of the list.
B. `fun` segfaults on lists of odd length.
C. `fun` prints every other element of the list.
D. [Correct Answer] [Your Answer] `fun` prints the elements of the list from `head` to the end.
E. None of the other options is correct.

{{{questionNumber}}}. In a doubly linked list of size n, you are given the address of the last node. What will be the time required to access the data stored in the second last node?

A. O(n)
B. It cannot be accessed
C. [Correct Answer] [Your Answer] O(1)
D. O(log n)
E. O(log log n)