

Practice questions for Linked List and Recursion

Assume the structure of a Linked List node is as follows.

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

Explain the functionality of following C functions.

1. What does the following function do for a given Linked List?

```
void fun1(struct node* head)
{
    if(head == NULL)
        return;

    fun1(head->next);
    printf("%d ", head->data);
}
```

fun1() prints the given Linked List in reverse manner. For Linked List 1->2->3->4->5, fun1() prints 5->4->3->2->1.

2. What does the following function do for a given Linked List ?

```
void fun2(struct node* head)
{
    if(head== NULL)
        return;
    printf("%d ", head->data);

    if(head->next != NULL )
        fun2(head->next->next);
    printf("%d ", head->data);
}
```

fun2() prints alternate nodes of the given Linked List, first from head to end, and then from end to head. If Linked List has even number of nodes, then fun2() skips the last node. For Linked List 1->2->3->4->5, fun2() prints 1 3 5 3 1. For Linked List 1->2->3->4->5->6, fun2() prints 1 3 5 3 1.

Below is a complete running program to test above functions.

```
#include<stdio.h>
#include<stdlib.h>

/* A linked list node */
struct node
{
    int data;
    struct node *next;
};

/* Prints a linked list in reverse manner */
void fun1(struct node* head)
{
    if(head == NULL)
        return;
```

```

    fun1(head->next);
    printf("%d  ", head->data);
}

/* prints alternate nodes of a Linked List, first
   from head to end, and then from end to head. */
void fun2(struct node* start)
{
    if(start == NULL)
        return;
    printf("%d  ", start->data);

    if(start->next != NULL )
        fun2(start->next->next);
    printf("%d  ", start->data);
}

/* UTILITY FUNCTIONS TO TEST fun1() and fun2() */
/* Given a reference (pointer to pointer) to the head
   of a list and an int, push a new node on the front
   of the list. */
void push(struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node =
        (struct node*) malloc(sizeof(struct node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Drier program to test above functions */
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;

    /* Using push() to construct below list
       1->2->3->4->5 */
    push(&head, 5);
    push(&head, 4);
    push(&head, 3);
    push(&head, 2);
    push(&head, 1);

    printf("\n Output of fun1() for list 1->2->3->4->5 \n");
    fun1(head);

    printf("\n Output of fun2() for list 1->2->3->4->5 \n");
    fun2(head);

    getchar();
    return 0;
}

```