Assignment-1

1/. Find the complexity of the below recurrence.

(i) $T(n) = \begin{cases} 3T(n-1) & , \text{ if } n>0 \\ 1 & \text{otherwise} \end{cases}$

Let us solve this using substitution.

$T(n) = 3T(n-1)$
$= 3^2 T(n-2)$
$= 3^3 T(n-3)$
$\vdots$
$= 3^n T(n-n)$
$= 3^n T(0)$
$= 3^n$

$\therefore$ complexity of the function is 3^n.

(ii) $T(n) = \begin{cases} 2T(n-1) - 1 & , \text{ if } n>0, \\ 1, & \text{otherwise} \end{cases}$

Let us solve this using substitution.

$T(n) = 2T(n-1) - 1$
$= 2(2T(n-2) - 1) - 1$
$= 2^2 T(n-2) - 2 - 1$

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} - \cdots 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} - \cdots 2^2 - 2^1 - 2^0$

$= 2^n - (2^n - 1)$

$\because (2^{n-1} + 2^{n-2} + \cdots 2^0 = 2^n - 1)$

$\underline{\underline{T(n) = 1}}$

$\therefore O(1)$ will be Time complexity of the function

2. Discuss Towers of Honoi puzzle.

→ Tower of Honoi is a mathematical puzzle where we have three rods (A, B and C) and N disks. Initially, all the disks are stacked in decreasing value of diameter, i.e., the smallest disk is placed on the top and they are on rod A.

The main objective of the puzzle is to move the entire stack to another rod (rod C), by following the specified conditions below:

- Only one disk can be moved at a time.
- Each move consists of taking the upper disk from one of the stacks and placing it on top

T.C: O(2^n)

```
int main()
{
  int n= 4;
  tower of Hanoi (n, A, C, B);
  return 0;
 }
```

Output:

```
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 3 from rod A to rod B
Move disk 1 from rod C to rod A
Move disk 2 from rod C to rod B
Move disk 1 from rod A to rod B
Move disk 4 from rod A to rod C
Move disk 1 from rod B to rod C
Move disk 2 from rod B to rod A
Move disk 1 from rod C to rod A
Move disk 3 from rod B to rod C
Move disk 1 from rod A to rod B
Move disk 2 from rod A to rod C
Move disk 1 from rod B to rod C
```

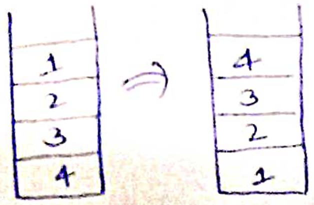
3/. How will you display a linked list from the end?

→ Given a pointer to the head node of a linked list, the task is to reverse linked list, changing links between nodes.

```
        struct Node* next = NULL;
    while (current != NULL) {
            next=curr->next;
            curr->next=prev;
            prev=curr;
            curr=next;
        }
     *head-ref = prev;
    }
```

4/. Given a stack, how to reverse the elements of the stack using only stack operations (push & pop)?

→ The idea of the solution is to hold all values in function call stack until it becomes empty. When stack becomes empty, insert all held items one by one at the bottom of the stack.



Algorithm

- Create a stack and push all the elements in it.
- Call reverse (), which will pop all the elements from the stack and pass the popped element to function bottom ().

of onother stock.

- No disk may be ploced on top of a smaller disk.

e.g.,

Input: 2
Output:

```
 Disk 1 moved from A to B
 Disk 2 moved from A to C
 Disk 1 moved from B to C
```

Using Recursion:- for three rods A, B & C.

- Shift 'N-1' disks from 'A' to 'B', using C.
- Shift last disk from 'A' to 'C'.
- Shift 'N-1' disks from 'B' to 'C' using C.

C program

```
void towerofHanoi (int n ,char A, char B, char C)
    {
      if (n==1)
      {
        printf("\n Move disk 1 from rod %c to rod
               %c", A, B);
        return;
      }
      towerofHanoi (n-1, A, C, B);
      printf ("\n Move disk %d from rod %c to rod
             %c , n, A, B);
      towerofHanoi (n-1, C, B, A);
    }
```

e.g.,
Head
1 -> 2 -> 3 -> 4 -> NULL
Required Output
4 -> 3 -> 2 -> 1 -> NULL

Algorithm

- Initialize three pointers prev as NULL, curr as head, and next as NULL.
- Iterate through linked list.
  - ↳ Before changing the next of curr, store the next node
    - next = curr->next
  - ↳ Now update the next pointer of curr to the prev
    - curr-> next = prev
  - ↳ Update prev as curr and curr as next
    - prev = curr
    - curr = next

C program

```
struct Node {
        int  data;
        struct Node* next;

   };
static void reverse (struct Node** head_ref)
{
    struct Node* prev = NULL;
    struct Node* curr = * head_ref ;
```

- Whenever bottom() is called, it'll insert the passed element at the bottom of stack.
- Print the stack.

C program

```
struct  sNode {
        char data;
        struct sNode*next;
};

void bottom (struct sNode**top-ref, int item)
```