INSTITUTE OF INFORMATION TECHNOLOGY

JAHANGIRNAGAR UNIVERSITY

Number of Assignment :	01
Course Tittle	**:** Algorithm Analysis and Design
Course Code	**:** ICT – 2201
Submission Date	**:** 24/01/2022

Submitted To	**Submitted By**
Dr. M. Abu Yousuf	Md. Shakil Hossain
Professor	Roll – 2023
IIT – JU	2nd year 2st Semester
	IIT – JU

Answer to the question no - 1

An algorithm to reverse the element of Stack using Stack operation.

Given stack is s and s_1, S_2 are two additional Stack.

Algorithm:

Step 1: Pop() each element from s and Push it into S_1 till S is empty

Step 2: Pop() each element from S_1 and Push () it into S_2 till S_1 is empty

Step 3: Pop () each element from S_2 and Push () it into s.

Now S Contain elements in reverse order.

Example:

let $S = \{2, 4, 6, 8, 10\}$

After Pop from S and Push into S_1.

$S_1 = \{10, 8, 6, 4, 2\}$

After Pop from S_1 and Push in S_2

$S_2 = \{2, 4, 6, 8, 10\}$

Finally Pop from S_2 and Push into S

$S = \{10, 8, 6, 4, 2\}$

Answer to the question no-2

For $n=1$, it means thnre is only one intenal node so it must have exactly K childern

K leaves

$$(K-1)n+1 = (K-1) \ 1+1 = K$$

Therefore for $n=1$ the above Propostion is true.

Assume that it is true for n.... if there are n internal nodes the number of leaves will be $(K-1)n+1$;

Now Consider the case that there are $n+1$ internal nods which mians any of the $(K+1)n+1$ leaves of the tree having n internal nods must sprout K leaves.

Now total number of leaves are

$([k-1]\,n)$ [out of $(k-1)\,n+1$ ... one lefe become an interanal node So remaining are $(k-1)n + K$ [new leaves sprouted from a previous leaf]

$$\text{Total number of Leaves} = [\,(k-1]\,n\,]+K$$
$$= (k-1)(n+1) + (k - (k-1))$$
$$= (k-1)\,(n+1)+1$$

Therefore a full k-array tree having $n+1$ internal nodes must have $(k-1)(n+1)+1$ leaves.

Answer to the Question no - 3

~~Given,~~

Algorithm: There are two function in thic method one is to Print all nodes at a given level Print current level ( ) and another one is Print level orden ( ) for relaice traverse.

```
Print Level order (tree)
        for i = height (tree) to 1 ;
            Printcurrent level (tree, i);

Print current level  (tree, level)
    if tree is Null the return ;
    if level is 1 then
            Print (tree → data);

    else if
        level > 1 then.

       Printcurrentlevel (tree → left, level-1);
       Printcurrenllevel (tree → righ, level-1);
```

Answer to thequestion no - 4

An Algorithm to find Sum of all nodes of a Binary Search Tree (BST)

Angorithm:

Step 1: Define a Class having all 3 ributes

```
class { data,
        lest,
        right }
```

Step 2: Wchen a node is created data will pass to the data attribute both left and right null.

```
{ node -> data = Key;
  node -> left = Null;
  node -> right = Null; }
```

Step 3: Define a Class has an attribute root

```
Class { root = Null;
      }
```

Step 4: ~~dat~~calculate Sum() will calculate Sume of all nodes.

1. Check wheather root = Null
   if root is null return 0;

2. if Not then
   return (root.key + calculate sum (root.left) + calculate sum (root.right))

Step 5: Print Sum;

THE END