LAB-02:-

K th largest sum in Binary tree.

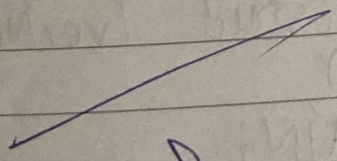
```
typedef struct TreeNode node;
int get_height(node *root)
{
    if (!root) return 0;
    int l = get_height(root->left)+1;
    int r = get_height(root->right)+1;
    return l>r ? l:r;
}
void get_level_sum(node *root, int level, long long *data)
{
    if (!root) return;
    data[level] += root->val;
    get_level_sum(root->left, level+1, data);
}
void print_arr(long long *arr, int n)
{
    for(int i=0; i<n; i++)
        printf("%lld", arr[i]);
    printf("\n");
}
void swap(long long *a, long long *b)
{
    long long t = *a;
    *a = *b;
    *b = t;
}
int partition(long long *data, long long *b, int l,i
{
    long long t = *a;
    *a = *b;
```

```
    *b=t;
}
int
    int i=l, j=l;
    long long pivot = data[h-1];
    for(j=l; j<h-1; j++)
        if(data[j]<pivot)
            swap(&data[i++], &data[j]);
    swap(&data[i], &data[h-1]);
    return i;
}
void quick_select(long long *data, int l, int h, int k)
{
    if (l+1 < h)
    {
        int p = partition(data, l, h);
        if (p==k) return;
        else if (p<k)
            quick_select(data, p+1, h, k);
        else
            quick_select(data, l, p, k);
    }
}
long long kthLargestLevelSum(struct TreeNode* root, int k)
{
    int n = get_height(root);
    if (k>n) return -1;
    long long val, *data = (long long*)calloc(n,
        sizeof(long long));
    get_level_sum(root, 0, data);
    quick_select(data, 0, n, n-k);
    return data[n-k];
}
```

=> outputs:-

case 1:
root = [5, 8, 9, 2, 1, 3, 7, 4, 6]
k = 2
output: 13

case 2:
root = [1, 2, null, 3]
k = 1
output: 3



16/5/24