**1.**class Solution {

public:

int knapSack(int capacity, vector<int> &val, vector<int> &wt) {

int n = val.size();

vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= capacity; j++) {

if (wt[i - 1] <= j) {

dp[i][j] = max(dp[i - 1][j], val[i - 1] + dp[i - 1][j - wt[i - 1]]);

} else {

dp[i][j] = dp[i - 1][j];

}

}

}

return dp[n][capacity];

}

};  
  
**2.Floor in sorted array**

class Solution {

public:

int findFloor(vector<int>& arr, int k) {

int n = arr.size();

int l = 0, h = n - 1;

while (l <= h) {

int m = l + (h - l) / 2;

if (arr[m] == k) {

return m;

} else if (arr[m] < k) {

l = m + 1;

} else {

h = m - 1;

}

}

return h >= 0 ? h : -1;

}

};

**3. Check Equal Arrays**

class Solution {

public:

bool check(vector<int>& arr1, vector<int>& arr2) {

sort(arr1.begin(), arr1.end());

sort(arr2.begin(), arr2.end());

return arr1 == arr2;

}

};

**4. Palindrome Linked List**

struct Node {

int data;

struct Node \*next;

Node(int x) {

data = x;

next = NULL;

}

};

class Solution {

public:

bool isPalindrome(Node \*head) {

Node\* currentNode=head;

stack<int>st;

while(currentNode!=NULL){

st.push(currentNode->data);

currentNode=currentNode->next;

}

while(head!=NULL){

int c=st.top();

st.pop();

if(head->data!=c){

return false;

}

head=head->next;

}

return true;

}

};  
  
**5. Balanced Tree Check**  
  
struct Node

{

int data;

struct Node\* left;

struct Node\* right;

Node(int x){

data = x;

left = right = NULL;

}

};

class Solution{

public:

int isBalRec(Node\* root)

{

if (root == NULL)

return 0;

int lh = isBalRec(root->left);

if (lh == -1)

return -1;

int rh = isBalRec(root->right);

if (rh == -1)

return -1;

if (abs(lh - rh) > 1)

return -1;

else

return max(lh, rh) + 1;

}

bool isBalanced(Node \*root)

{

return (isBalRec(root) > 0);

}

};

**6. Triplet Sum in Array**class Solution {

public:

bool find3Numbers(int arr[], int n, int x) {

std::sort(arr,arr+n);

for(int i=0;i<n-2;i++){

int left=i+1;//index of first element among the rem elements

int right=n-1;//index of last ele

while(left<right){

int current\_sum=arr[i]+arr[left]+arr[right];

if(current\_sum==x){

return 1;

}

else if(current\_sum<x){

left++;

}

else{

right--;

}

}

}

return 0;

}

};