Assignment Solutions | Binary search - 1 | Week 10

Given a sorted array of n elements and a target ‘x’. Find the last occurrence of ‘x’ in the array. If
‘x’ does not exist return -1.
Input 1: arr[] = {1,2,3,3,4,4,4,5} , x = 4
Output 1: 6
Solution:

```
#include<bits/stdc++.h>
using namespace std;
int lastOccurrence(vector<int>&a , int tgt) {
int low = 0 , high = a.size() - 1;
int answer = -1;
while(low <= high){
int mid = low + (high - low)/2;
if(a[mid] <= tgt){
answer = mid;
low = mid + 1;
}
else high = mid - 1;
}
return answer;
}
int main(){
int n;
cin>>n;
vector<int>a(n);
for(int i=0;i<n;i++)cin>>a[i];
int tgt;
cin>>tgt;
```

Given a sorted binary array, efficiently count the total number of 1’s in it.
Input 1 : a = [0,0,0,0,1,1]

Output 1: 2
Solution:

```
#include <bits/stdc++.h>
using namespace std;
int firstOccurrence(vector<int>&a , int n , int tgt){
int low = 0 , high = n - 1;
int ans = -1;
while(low <= high){
int mid = low + (high - low)/2;
if(a[mid] == tgt){
```

```
ans = mid;
high = mid - 1;
}
else low = mid + 1;
}
return low;
}
int main() {
int n;
cin>>n;
vector<int>a(n);
for(int i=0;i<n;i++)cin>>a[i];
cout<<n - firstOccurrence(a , n , 1);
```

Given a matrix having 0-1 only where each row is sorted in increasing order, find the row with the
maximum number of 1's.
Input matrix : 0 1 1 1
0 0 1 1
1 1 1 1 // this row has maximum 1s
0 0 0 0
Output: 2
Solution:

```
#include<bits/stdc++.h>
using namespace std;
int first(vector<int>&arr, int low, int high){
if(high >= low){
// Get the middle index
int mid = low + (high - low)/2;
// Check if the element at middle index is first 1
if ( ( mid == 0 || arr[mid-1] == 0) && arr[mid] == 1)
return mid;
// If the element is 0, recur for right side
else if (arr[mid] == 0)
return first(arr, (mid + 1), high);
// If element is not first 1, recur for left side
else
return first(arr, low, (mid -1));
}
return -1;
}
int rowWithMax1s(vector<vector<int>>&a){
// Initialize max values
```

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n]
inclusive in sorted order.

There is only one repeated number in nums, return this repeated number.
Input 1: arr[] = {1,2,3,3,4}
Output 1: 3
Input 2: arr[] = {1,2,2,3,4,5}
Output 2: 2
Solution:

```
#include<bits/stdc++.h>
using namespace std;
int findDuplicate(vector<int>& nums) {
int low = 1, high = nums.size() - 1, cnt;
while(low <= high){
int mid = low + (high - low) / 2;
cnt = 0;
// cnt number less than equal to mid
for(int n : nums)
{
if(n <= mid)
++cnt;
}
// binary search on left
if(cnt <= mid)
low = mid + 1;
else
// binary search on right
high = mid - 1;
}
return low;
```

Given a number ‘n’. Predict whether ‘n’ is a valid perfect square or not.
Input 1: n = 36
Output 1: yes
Input 2: n = 45

Output 2: no
Solution:

```
#include<bits/stdc++.h>
using namespace std;
bool isPerfectSquare(int num) {
int low = 1 , high = num;
while(low <= high){
long long int mid = low + (high - low)/2;
if((long long int)mid*mid == num)return true;
else if((long long int)mid*mid < num)low = mid + 1;
else high = mid - 1;
}
return false;
```

```
}
int main(){
int n;
cin>>n;
isPerfectSquare(n) ? cout<<"Yes" : cout<<"No";
}
```

You have n coins and you want to build a staircase with these coins. The staircase consists of k
rows where the ith row has exactly i coins. The last row of the staircase may be incomplete.

Given the integer n, return the number of complete rows of the staircase you will build.
Example 1:
Input: n = 5
Output: 2
Explanation: Because the 3rd row is incomplete, we return 2.
Example 2:
Input: n = 8
Output: 3
Explanation: Because the 4th row is incomplete, we return 3.
Solution :

```
#include<bits/stdc++.h>
using namespace std;
int arrangeCoins(int n) {
long low = 0; // we use "long" because we may get an integer overflow
long high = n;
while(low <= high){
long mid = low + (high - low) / 2;
long coinsUsed = mid * (mid + 1)/2;
if(coinsUsed == n){
return (int)mid;
}
if(n < coinsUsed){
high = mid - 1;
}
else{
low = mid + 1;
}
}
return (int)high; // cast as an "int" because it was initialized as a "long"
}
int main(){
int n;
cin>>n;
```