PROBLEM 1-

BINARY SEARCH(LEET CODE)

class Solution {

public:

int search(vector<int>& nums, int target) {

int pivot , left=0, right=nums.size()-1;

while(left <=right)

{

pivot=left +(right-left)/2;

if(nums[pivot]==target)

return pivot;

if(target<nums [pivot])

right=pivot-1;

else

left=pivot+1;

}

return-1;

}

};

PROBLEM 2 - FIRST BAD VERSION

class Solution {

public:

int firstBadVersion(int n) {

long long int beg,last,mid;

int lowest\_bad = n;

int highest\_true = 0;

while (highest\_true != lowest\_bad - 1) {

// preventing int overflow

int check = lowest\_bad + (highest\_true - lowest\_bad) / 2;

if (!isBadVersion(check)) {

highest\_true = check;

} else {

lowest\_bad = check;

}

}

return lowest\_bad;

}

};

PROBLEM 3 SEARCH INSERTION POSITION

class Solution {

public:

int searchInsert(vector<int>& nums, int target) {

int start = 0;

int end = nums.size() - 1;

while(start <= end){

int mid = start + (end - start)/2;

if (nums[mid] == target)

return mid;

else if (nums[mid] < target)

start = mid + 1;

else

end = mid - 1;

}

return start;

}

int main() {

vector<int> vec = {1, 2, 4, 5, 6, 7};

int target = 3;

int index = searchInsert(vec, target);

cout << "Index: " << index;

return 0;

}

};