|  |  |
| --- | --- |
| **WORKSHEET-1.3** | |
| **Student Name: Shubham Kumar** | **UID: 21BCS7104** |
| **Branch: CSE** | **Section/Group: 614-A** |
| **Semester: 6th** | **Date of Performance: 31-01-2024** |
| **Subject Name: Advanced Programming Lab-2** | **Subject Code: 21CSP-351** |

**1. Aim:** Heap model.

A) Return the weight of the last remaining stone. If there are no stones left, return 0.( Last Stone Weight)

B) Return the furthest building index (0-indexed) you can reach if you use the given ladders and bricks optimally.( Furthest Building You Can Reach)

**2. Source Code/Output:**

**A)**

class Solution {

public:

int lastStoneWeight(vector<int>& stones) {

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

while(stones.size() > 1){

int a = stones.back();

stones.pop\_back();

int b = stones.back();

stones.pop\_back();

if(a != b){

stones.push\_back(a - b);

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

}

}

if(stones.size()) {

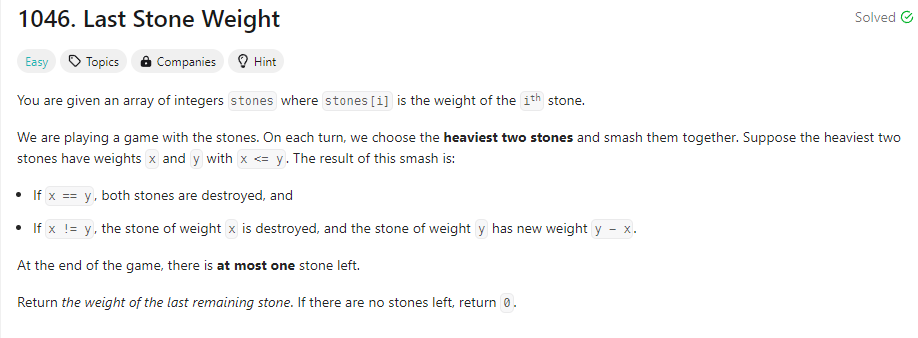
return stones[0];

}

return 0;

}

};





B)

class Solution {

public:

int furthestBuilding(vector<int>& heights, int bricks, int ladders) {

priority\_queue<int, vector<int>, greater<int>> q;

int n = heights.size();

for (int i = 0; i < n - 1; ++i) {

int a = heights[i], b = heights[i + 1];

int d = b - a;

if (d > 0) {

q.push(d);

if (q.size() > ladders) {

bricks -= q.top();

q.pop();

if (bricks < 0) {

return i;

}

}

}

}

return n - 1;

}

};

