**Worksheet-2**

**Student Name:-** Pushpraj Roy **UID:-** 20BCS9866

**Branch:-** BE- CSE **Section/Group:-** WM\_617 “A”

**Subjetct Code:-** 20CSP-314 **Semester:-** 5th

**Subject Name:-** Competitive Coding Lab

1. **Aim/Overview of the practical:-**
2. Down to zero problem
3. Truck Tour Problem
4. **Task to be done/ Which logistics used:-**
5. You are given Q queries. Each query consists of a single number N. You can perform any of the two operations on N in each move:

1: If we take 2 integers a and b where ,N=a X b (a!=1,b!=1) , then we can change N =max(a,b)

2: Decrease the value of N by 1 .

Determine the minimum number of moves required to reduce the value of N to 0

1. Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered to 0 To (N - 1 ) (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Initially, you have a tank of infinite capacity carrying no petrol. You can start the tour at any of the petrol pumps. Calculate the first point from where the truck will be able to complete the circle. Consider that the truck will stop at each of the petrol pumps. The truck will move one kilo-meter for each litre of the petrol.

1. **Steps for experiment/practical/Code:**
2. **Down to zero problem**

#include <map>

#include <cmath>

#include <queue>

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

#define NSIZE 1000000

vector< vector<int> > ar(NSIZE+1);

bool primes[NSIZE+1];

int cache[NSIZE], dcache[NSIZE];

void gen\_primes()

{

for (int i = 2; i <= NSIZE; i++) {

}

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

bool prime = true;

for (int j = 2; j\*j<= i; j++) {

if (i % j == 0) {

prime = false;

break;

}

}

primes[i] = prime;

}

}

void get\_a(int n)

{

if (primes[n])

return;

if (dcache[n] != -1)

return;

else

dcache[n] = 1;

for (int i = 2; i\*i <= n; i++) {

if (n % i == 0) {

int v = n / i;

if (v == 1 || i == 1)

continue;

v = v > i ? v : i;

ar[n].push\_back(v);

}

}

}

int w[NSIZE];

void fill\_cache(int steps, int number, int start)

{

int indx = start;

int st = 1;

if (cache[start] != -1)

st = cache[start] + 1;

while(1) {

int pos = w[indx];

cache[pos] = st++;

indx = pos;

if (st == steps + 1)

break;

}

}

int \*q;

int qpos = 0;

int qend = 0;

int steps = 0;

int cal\_steps(int v)

{

for (int i = 0; i <= v; i++)

w[i] = -1;

qpos = 0;

qend = 0;

steps = 0;

q[qend++] = v;

q[qend++] = -1;

while(1) {

int val = q[qpos++];

if (val == -1) {

steps ++;

q[qend++] = -1;

val = q[qpos++];

}

if (val == 0) {

return steps;

}

get\_a(val);

for (int i = 0; i < ar[val].size(); i++) {

if (w[ar[val][i]] == -1)

w[ar[val][i]] = val;

int tmp\_val = ar[val][i];

q[qend++] = tmp\_val;

}

val -= 1;

q[qend++] = val;

}

return -1;

}

int main() {

std::ios\_base::sync\_with\_stdio (false);

q = new int[NSIZE \* 19];

int n, v;

cin >> n;

int max = n;

for (int i = 0; i < NSIZE; i++) {

cache[i] = -1;

dcache[i] = -1;

}

gen\_primes();

while(n--) {

cin >> v;

if (v == 0) {

cout << "0" << endl;

continue;

}

cout << cal\_steps(v) << endl;

}

return 0;

}

1. **Truck Tour Problem**

#include <cstdio>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

int n, p[100000], d[100000];

int main() {

cin >> n;

for (int i = 0; i < n; ++i) scanf("%d%d", &p[i], &d[i]);

int ret = 0, amount = 0, sum = 0;

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

p[i] -= d[i];

sum += p[i];

if (amount + p[i] < 0) {

amount = 0;

ret = i + 1;

}

else amount += p[i];

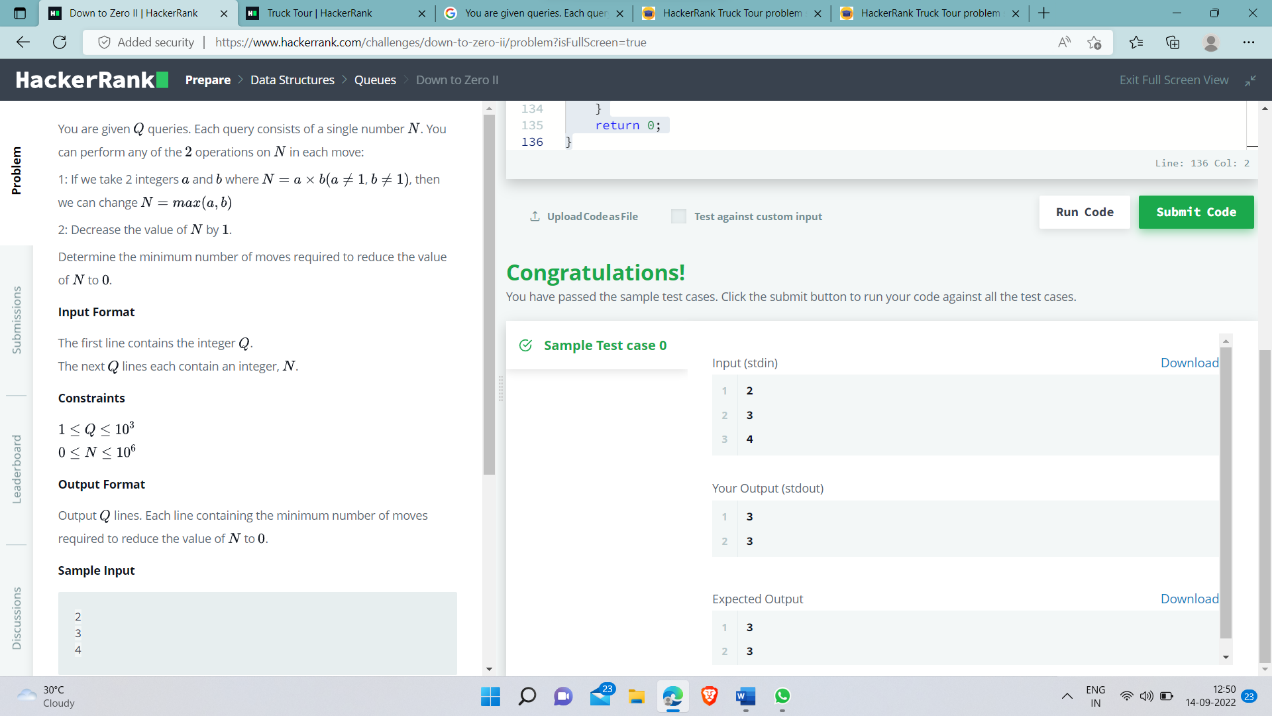
}

cout << (sum >= 0 ? ret : -1) << endl;

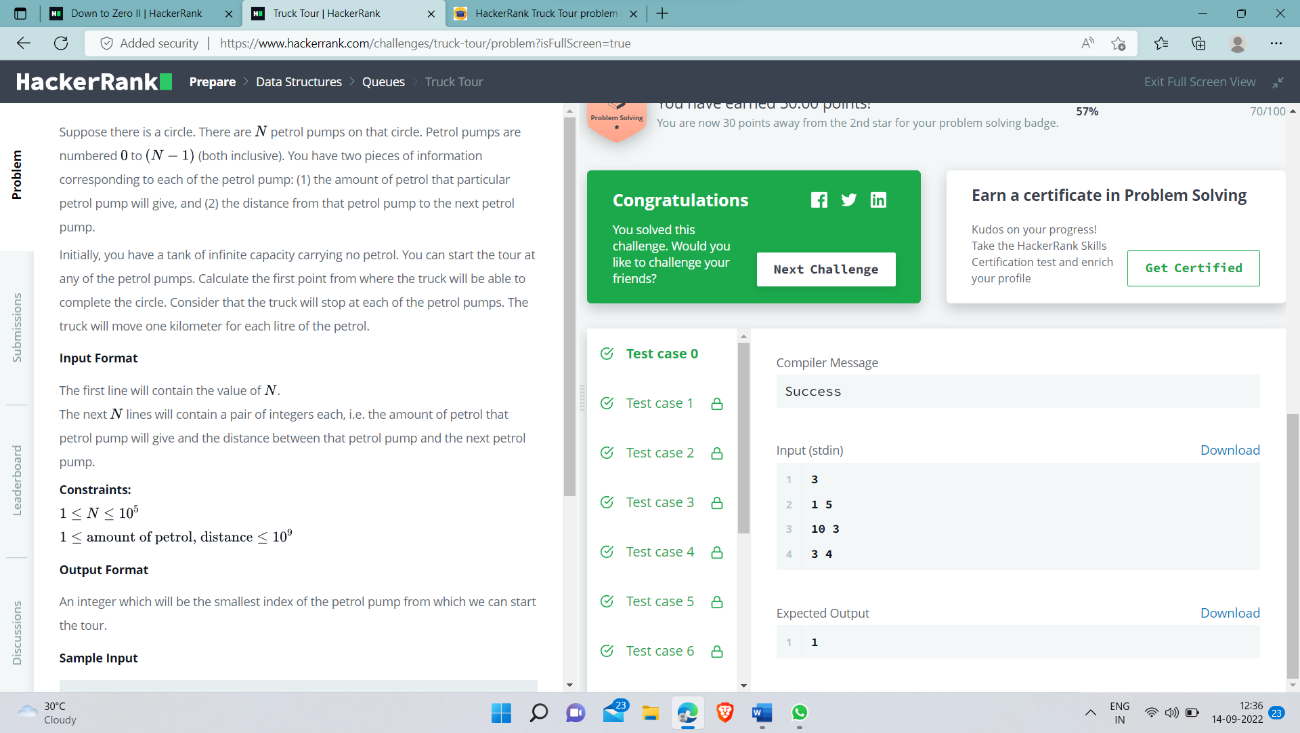
return 0;

}

1. **Result/Output/Writing Summary:**
2. **Down to zero problem**

****

1. **Truck Tour Problem**

****