# DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

**Experiment 3.2**

**Student Name:** Praduman Kumar

**Branch:** BE CSE

**Semester:** 6th

**Subject Name:** CC Lab

**UID:** 20BSC9446

**Section/Group:** 714/A

**Date of Performance:** 05/05/2023

**Subject Code:** 20CSP-351



## Aim/Overview of the practical:

**To implement the concept of backtracking.**

Binary Watch

A binary watch has 4 LEDs on the top to represent the hours (0-11), and 6 LEDs on the bottom to represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right.

https://leetcode.com/problems/binary-watch/

## Code:

class Solution {

public List<String> readBinaryWatch(int num) { List<String> result = new ArrayList<>();

for (int hh = 0; hh < 12; hh++)

for (int mn = 0; mn < 60; mn++) if (aux(hh, mn, num))

if (mn < 10) result.add(String.format("%d:0%d", hh, mn));

else

result.add(String.format("%d:%d", hh, mn)); return result;

}

private boolean aux(int hh, int mn, int num){ int temp = 0;

while(hh != 0 || mn != 0){ if (hh !=0 ){

temp += hh % 2; hh /=2;

}

if (mn != 0){

temp += mn % 2; mn /= 2;

}

}

return temp == num;

}

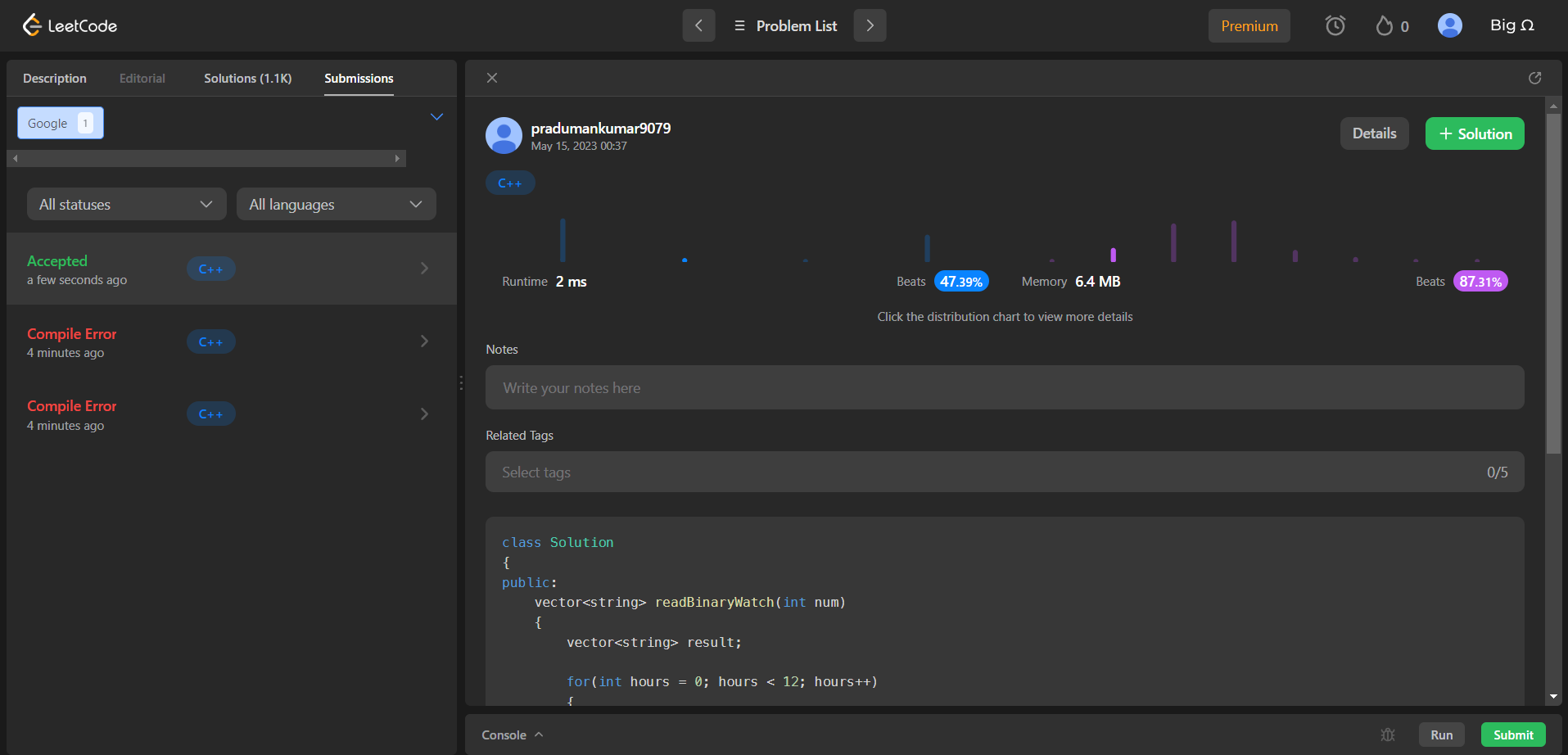
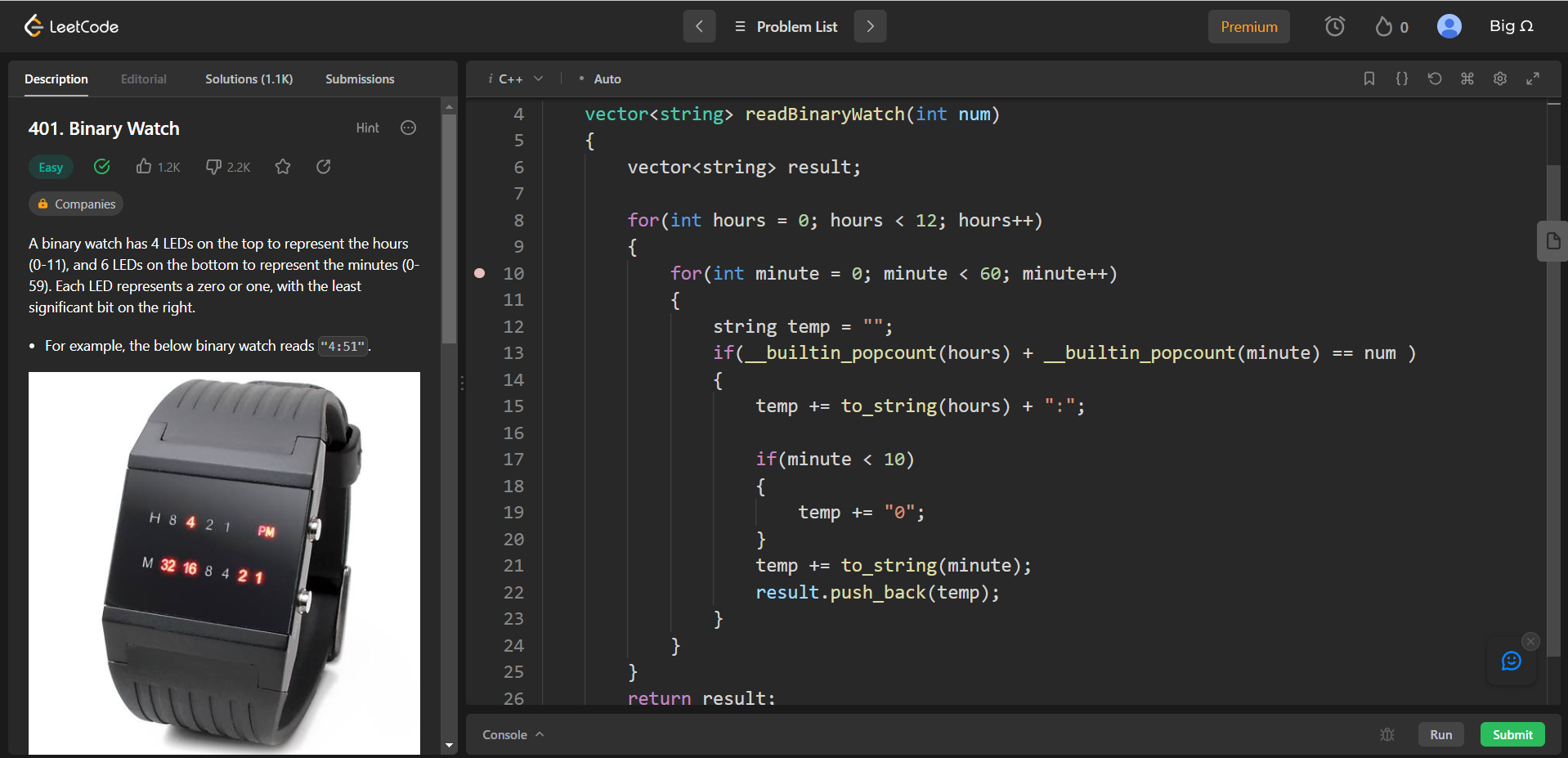
}



# DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

## Result/Output/Writing Summary:

****

1. **Aim/Overview of the practical:**

Stickers to Spell Word

We are given n different types of stickers. Each sticker has a lowercase English word on it.

You would like to spell out the given string target by cutting individual letters from your collection of stickers and rearranging them. You can use each sticker more than once if you want, and you have infinite quantities of each sticker.

Return *the minimum number of stickers that you need to spell out* target. If the task is impossible, return -1.

https://leetcode.com/problems/stickers-to-spell-word/

## Code:

class Solution { public:

int minStickers(vector<string>& stickers, string target) { vector<vector<int>> sticker\_counts(stickers.size(), vector<int>(26)); unordered\_map<string, int> dp;

for (int i = 0; i < stickers.size(); ++i) { for (const auto& c : stickers[i]) {

++sticker\_counts[i][c - 'a'];

}

}

dp[""] = 0;

return minStickersHelper(sticker\_counts, target, &dp);

}

private:

int result = numeric\_limits<int>::max(); vector<int> target\_count(26);

for (const auto& c : target) {

++target\_count[c - 'a'];

}

for (const auto& sticker\_count : sticker\_counts) { if (sticker\_count[target[0] - 'a'] == 0) {

continue;

}

string new\_target;

for (int i = 0; i < target\_count.size(); ++i) { if (target\_count[i] - sticker\_count[i] > 0) {

new\_target += string(target\_count[i] - sticker\_count[i], 'a' + i);

}

}

if (new\_target.length() != target.length()) {

int num = minStickersHelper(sticker\_counts, new\_target, dp); if (num != -1) {

result = min(result, 1 + num);

} } }

(\*dp)[target] = (result == numeric\_limits<int>::max()) ? -1 : result; return (\*dp)[target];

}};

## Result/Output/Writing Summary:

## 

**Learning outcomes (What I have learnt):**

* Learned the concept of Backtracking.