**EXPERIMENT-5**

**Student Name:** Shubham Sharma **UID:**22BET10358

**Branch:** BE -IT **Section/Group:**22BET\_IOT-703(A)

**Semester:** 6th **Subject Code:** 22ITP-351

**PROBLEM-1**

**AIM:-**

Longest Increasing Subsequence

**CODE:-**

public class Solution {

    public int lengthOfLIS(int[] nums) {

        if (nums == null || nums.length == 0) {

            return 0;

        }

        int n = nums.length;

        int[] dp = new int[n];

        Arrays.fill(dp, 1);

        for (int i = 1; i < n; ++i) {

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

                if (nums[i] > nums[j]) {

                    dp[i] = Math.max(dp[i], dp[j] + 1);

                }

            }

        }

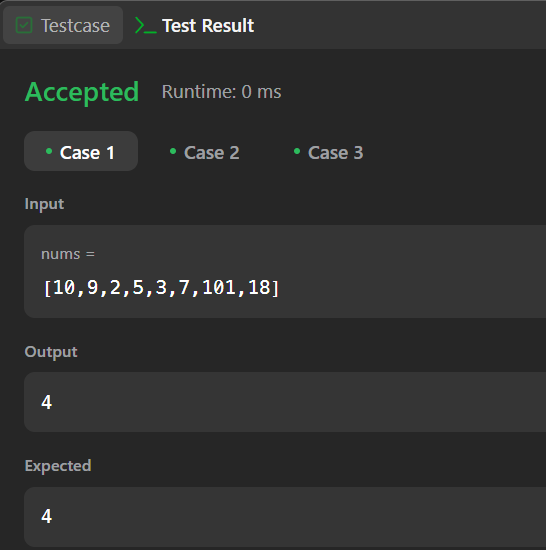
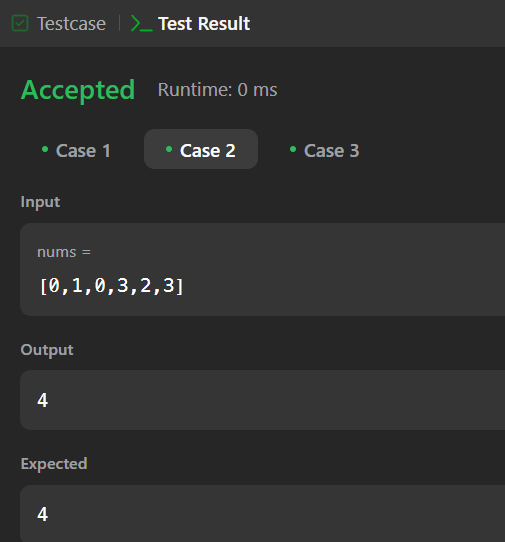
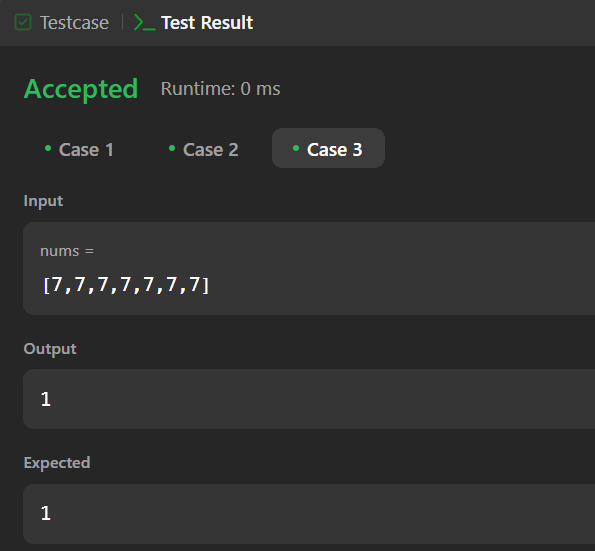
        int maxLength = Arrays.stream(dp).max().orElse(0);

        return maxLength;

    }

}

**OUTPUT:-**

**  **

**PROBLEM-2**

**AIM:-**

Maximum Product Subarray

**CODE:-**

class Solution {

    public int maxProduct(int[] nums) {

        int res = Integer.MIN\_VALUE;

        for (int n : nums) {

            res = Math.max(res, n);

        }

        int curMax = 1, curMin = 1;

        for (int n : nums) {

            int temp = curMax \* n;

            curMax = Math.max(temp, Math.max(curMin \* n, n));

            curMin = Math.min(temp, Math.min(curMin \* n, n));

            res = Math.max(res, curMax);

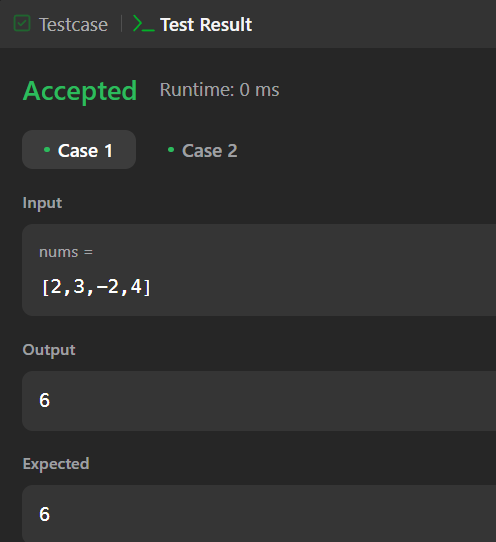
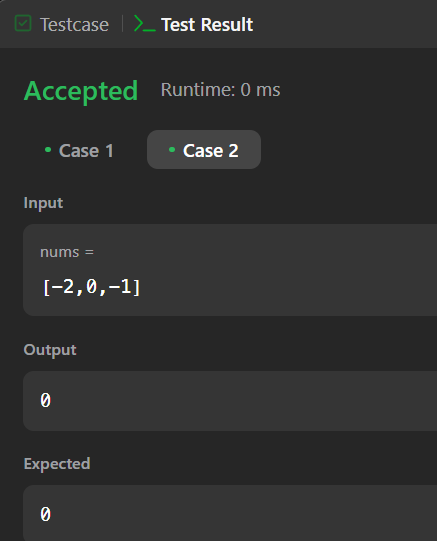
        }

        return res;

    }

}

**OUTPUT:**

** **

**PROBLEM-3**

**AIM:-**

Decode Ways

**CODE:-**

class Solution {

    public int numDecodings(String s) {

        if (s.charAt(0) == '0') {

            return 0;

        }

        int n = s.length();

        int[] dp = new int[n + 1];

        dp[0] = dp[1] = 1;

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

            int one = Character.getNumericValue(s.charAt(i - 1));

            int two = Integer.parseInt(s.substring(i - 2, i));

            if (1 <= one && one <= 9) {

                dp[i] += dp[i - 1];

            }

            if (10 <= two && two <= 26) {

                dp[i] += dp[i - 2];

            }

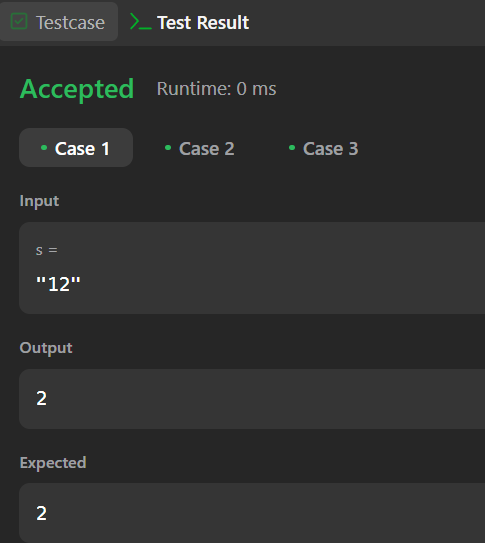
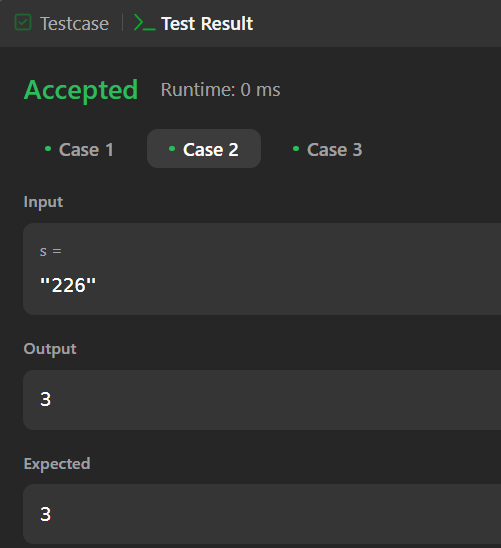
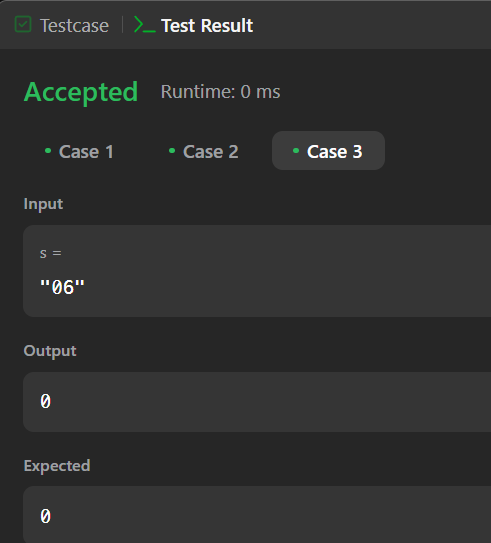
        }

        return dp[n];

    }

}

**OUTPUT:-**

**  **

**PROBLEM-4**

**AIM:-**

Best time to buy and Sell a Stock with Cooldown

**CODE:-**

class Solution {

public int maxProfit(int[] prices) {

int coolDown = 0, sell = 0, hold = Integer.MIN\_VALUE;

for (int stockPrice : prices) {

int prevCoolDown = coolDown, prevSell = sell;

coolDown = Math.max(prevCoolDown, sell);

sell = hold + stockPrice;

hold = Math.max(hold, prevCoolDown - stockPrice);

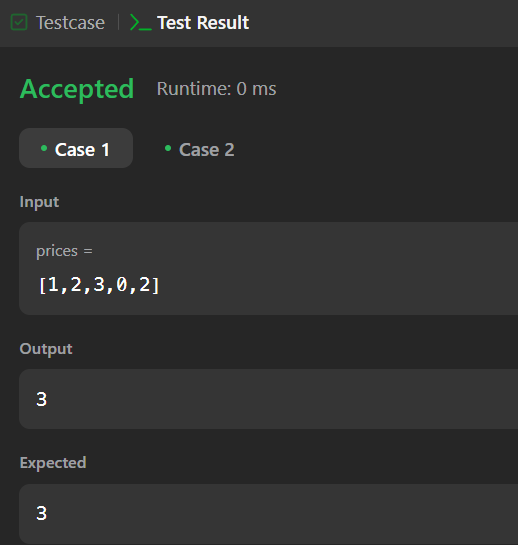
}

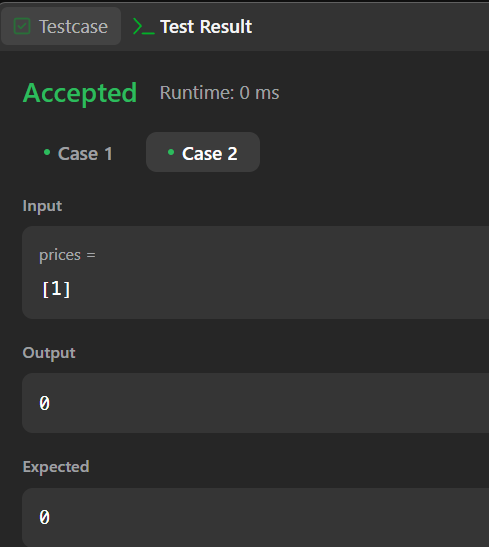
return Math.max(coolDown, sell);

}

}

**OUTPUT:-**

****

****

**PROBLEM-5**

**AIM:-**

Perfect Squares

**CODE:-**

class Solution {

public int numSquares(int n) {

int[] dp = new int[n + 1];

Arrays.fill(dp, Integer.MAX\_VALUE);

dp[0] = 0;

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

int min\_val = Integer.MAX\_VALUE;

for (int j = 1; j \* j <= i; ++j) {

min\_val = Math.min(min\_val, dp[i - j \* j] + 1);

}

dp[i] = min\_val;

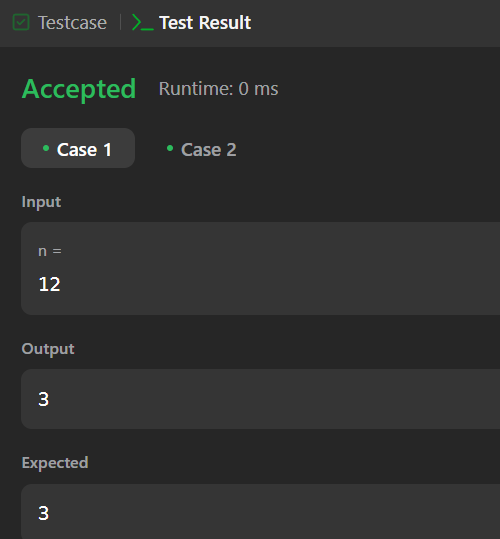
}

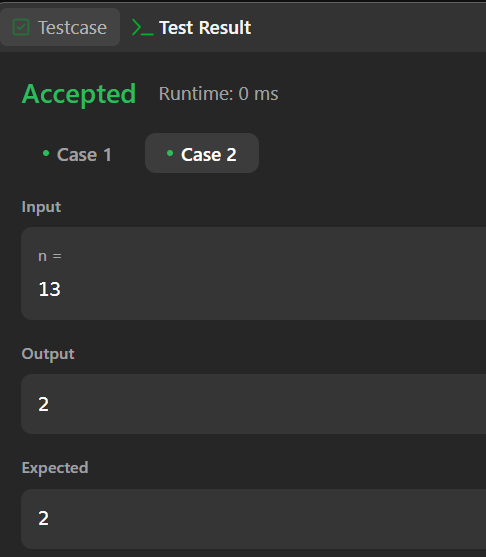
return dp[n];

}

}

**OUTPUT:-**

****

****

**PROBLEM-6**

**AIM:-**

Word Break

**CODE:-**

class Solution {

public boolean wordBreak(String s, List<String> wordDict) {

int n = s.length();

boolean[] dp = new boolean[n + 1];

dp[0] = true;

int max\_len = 0;

for (String word : wordDict) {

max\_len = Math.max(max\_len, word.length());

}

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

for (int j = i - 1; j >= Math.max(i - max\_len - 1, 0); j--) {

if (dp[j] && wordDict.contains(s.substring(j, i))) {

dp[i] = true;

break;

}

}

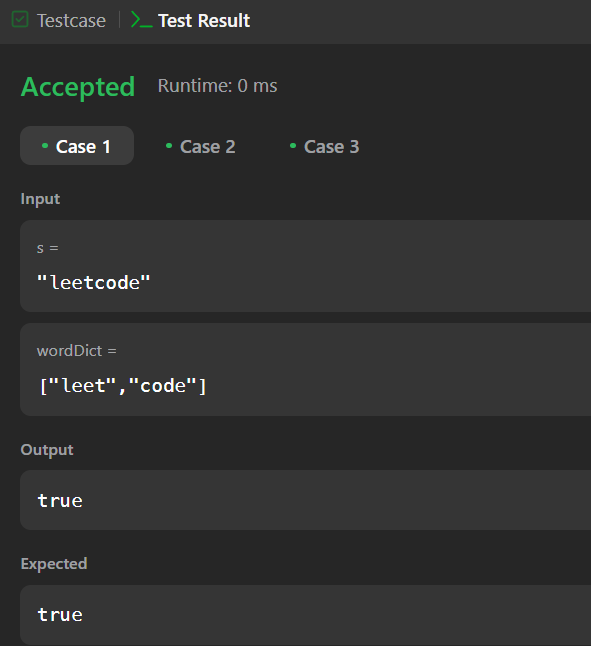
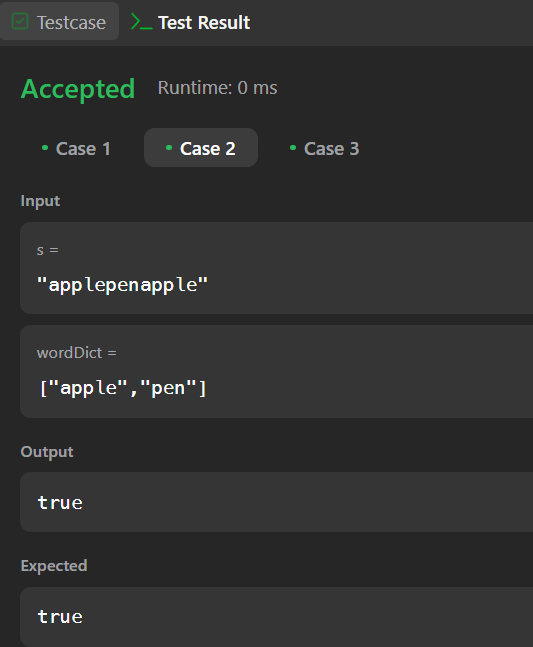
}

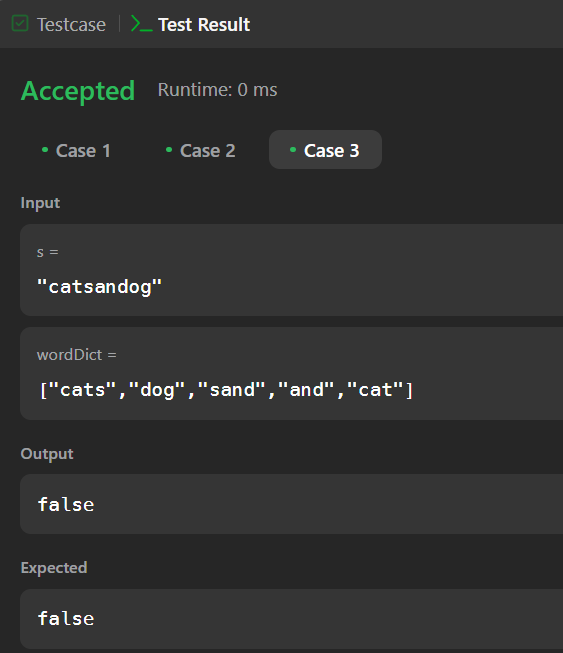
return dp[n];

}

}

**OUTPUT:-**

** **

****

**PROBLEM-7**

**AIM:-**

Word Break 2

**CODE:-**

class Solution {

    public List<String> wordBreak(String s, List<String> wordDict) {

        int n = s.length();

        Set<String> wordSet = new HashSet<>(wordDict);

        List<List<String>> dp = new ArrayList<>();

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

            dp.add(new ArrayList<>());

        }

        dp.get(0).add("");

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

            List<String> temp = new ArrayList<>();

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

                String suffix = s.substring(j, i);

                if (wordSet.contains(suffix)) {

                    for (String substring : dp.get(j)) {

                        temp.add(substring + (substring.isEmpty() ? "" : " ") + suffix);

                    }

                }

            }

            dp.set(i, temp);

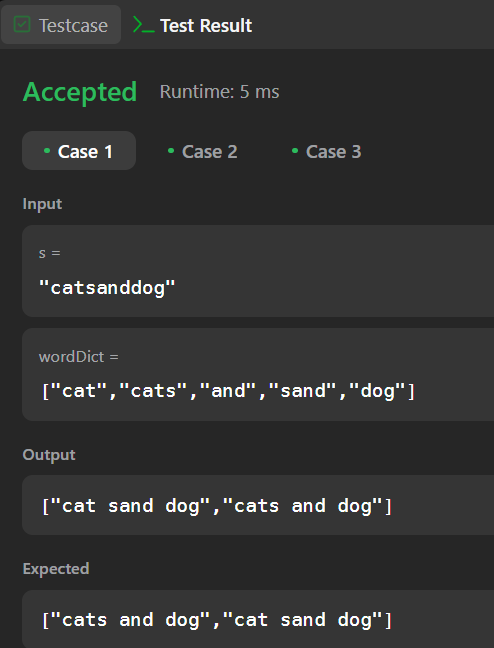
        }

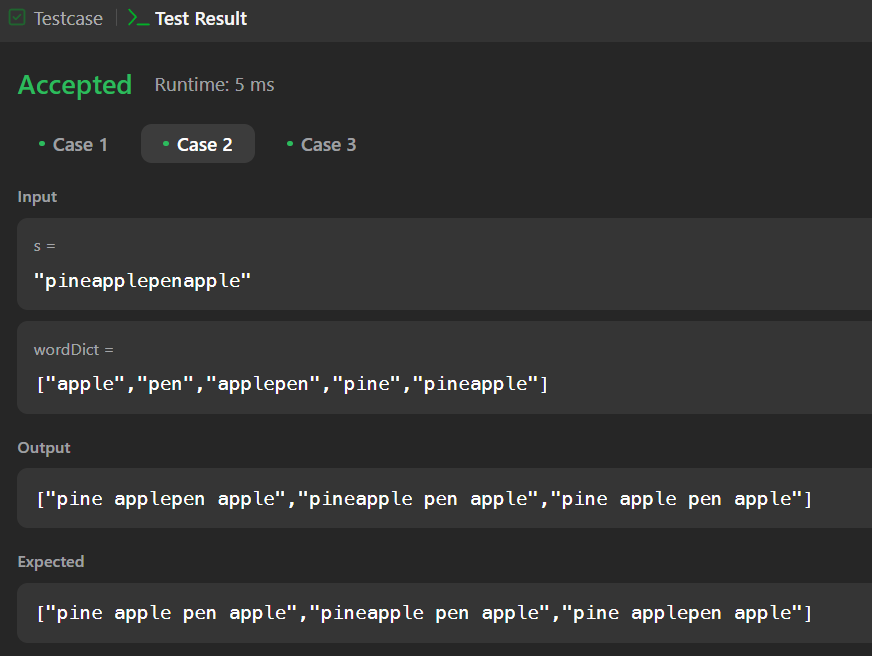
        return dp.get(n);

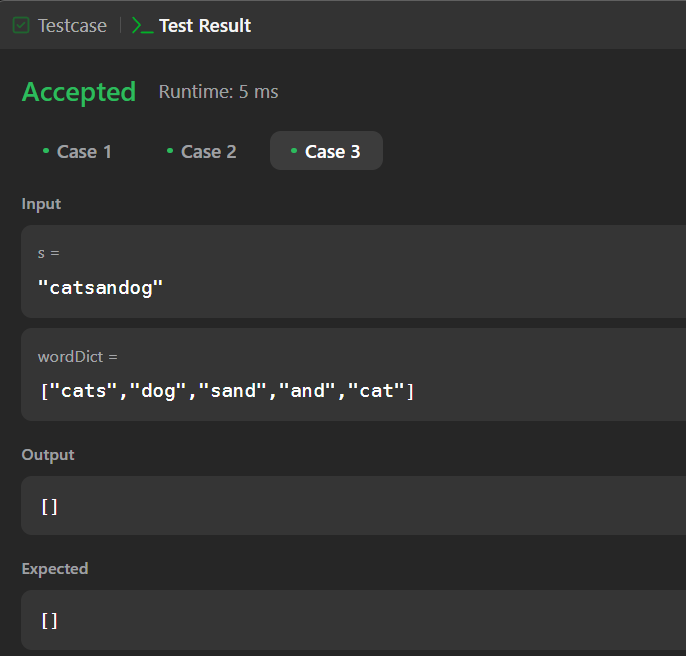
    }

}

**OUTPUT:-**

****

****

****