Experiment 5

Student Name: Arun **UID:** 22BET10320

Branch: Information Technology **Section/Group:** 22BET_IOT-701/A

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

Problem 1

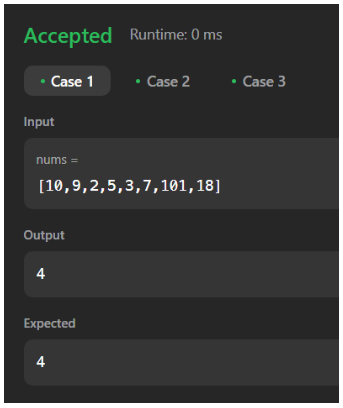
Aim:

Longest Increasing Subsequence

Code:

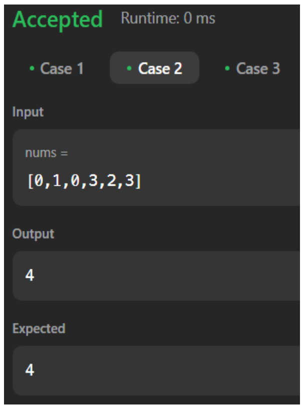
```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        vector<int> res;
        for (int n : nums) {
            if (res.empty() || res.back() < n) {
                res.push_back(n);
            } else {
                int idx = binarySearch(res, n);
                res[idx] = n;
            }
             return res.size();
    }private:
    int binarySearch(const vector<int>& arr, int target) {
        int left = 0;
        int right = arr.size() - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (arr[mid] == target) {
                return mid;
            } else if (arr[mid] > target) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return left;
    }
};
```

Output:



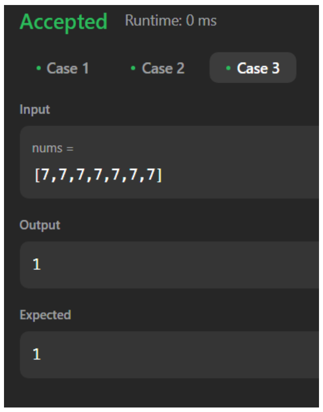
```
Accepted Runtime: 0 ms
Case 1  Case 2  Case 3
Input
nums =
[10,9,2,5,3,7,101,18]
Output
4
Expected
4
```

Case 1



```
Accepted Runtime: 0 ms
Case 1  Case 2  Case 3
Input
nums =
[0,1,0,3,2,3]
Output
4
Expected
4
```

Case 2



```
Accepted Runtime: 0 ms
Case 1  Case 2  Case 3
Input
nums =
[7,7,7,7,7,7,7]
Output
1
Expected
1
```

Case 3

Problem 2

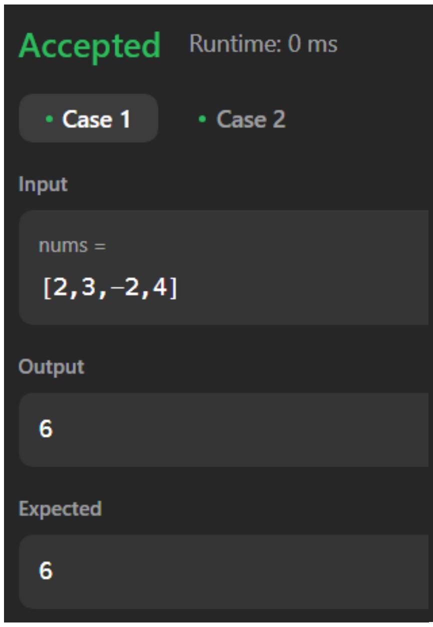
Aim:

Maximum Product Subarray

Code:

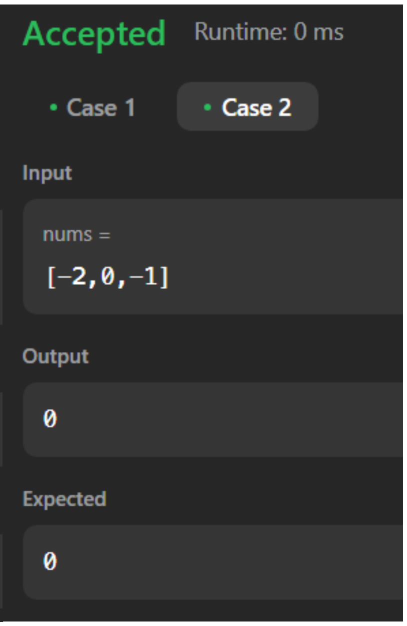
```
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int maxi = INT_MIN;
        int prod=1;

        for(int i=0;i<nums.size();i++)
        {
          prod*=nums[i];
          maxi=max(prod,maxi);
          if(prod==0)
           prod=1;
        }
        prod=1;
        for(int i=nums.size()-1;i>=0;i--)
        {
          prod*=nums[i];

          maxi=max(prod,maxi);
          if(prod==0)
           prod=1;
        }
        return maxi;
    }
};
```

Output:



```
Input
nums =
[2,3,-2,4]
Output
6
Expected
6
```

Test Case 1



```
Input
nums =
[-2,0,-1]
Output
0
Expected
0
```

Test Case 2

Problem 3

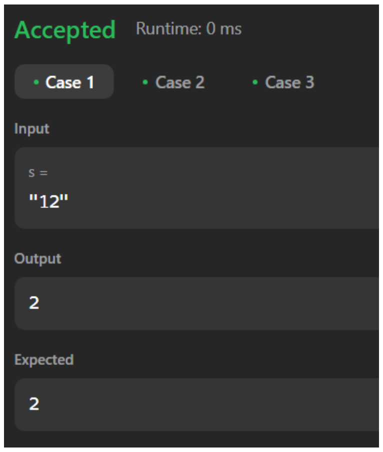
Aim:

Decode Ways

Code:

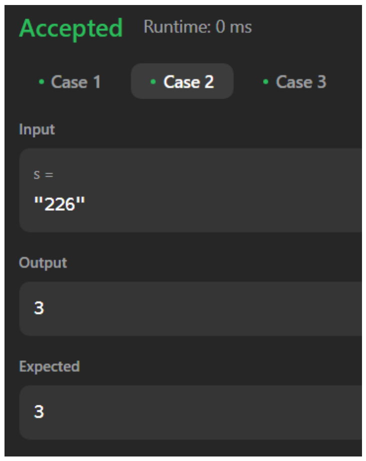
```
class Solution {
public:
    int numDecodings(string s) {
        if (s[0] == '0') {
            return 0;
        }

        int n = s.length();
        vector<int> dp(n + 1, 0);
        dp[0] = dp[1] = 1;

        for (int i = 2; i <= n; i++) {
            int one = s[i - 1] - '0';
            int two = stoi(s.substr(i - 2, 2));

            if (1 <= one && one <= 9) {
                dp[i] += dp[i - 1];
            }
            if (10 <= two && two <= 26) {
                dp[i] += dp[i - 2];
            }
        }

        return dp[n];
    }
};
```

Output:



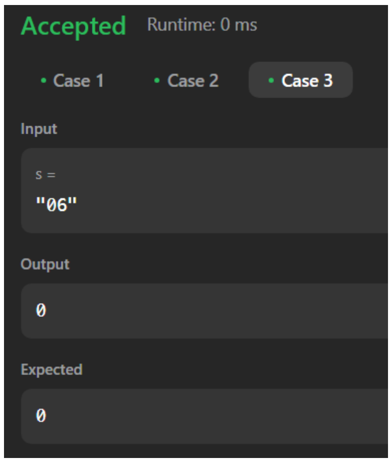
```
Accepted   Runtime: 0 ms
Case 1   Case 2   Case 3
Input
s =
"12"
Output
2
Expected
2
```

Case 1



```
Accepted   Runtime: 0 ms
Case 1   Case 2   Case 3
Input
s =
"226"
Output
3
Expected
3
```

Case 2



```
Accepted   Runtime: 0 ms
Case 1   Case 2   Case 3
Input
s =
"06"
Output
0
Expected
0
```

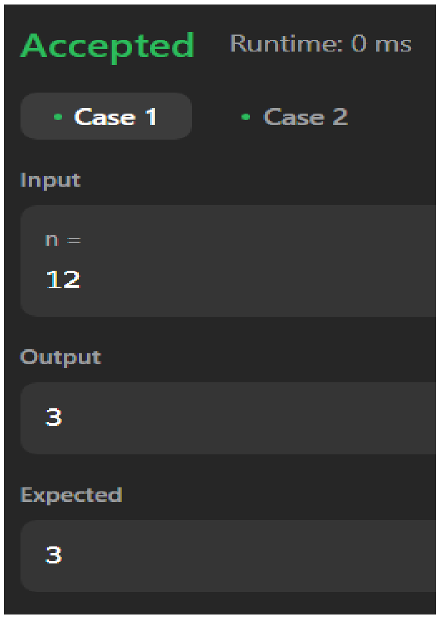
Case 3

Problem 4

Aim:
Perfect Squares

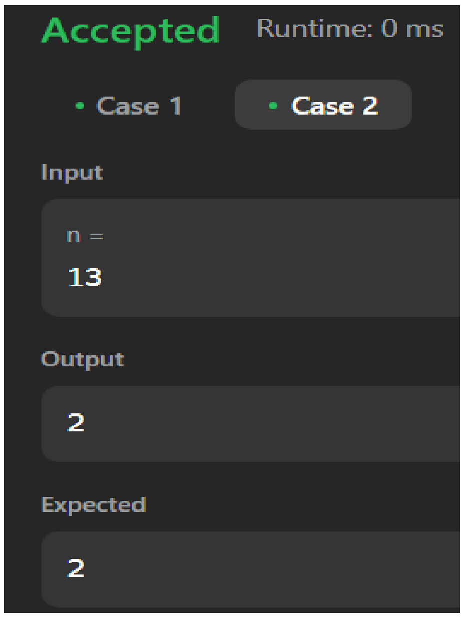
Code:
```
class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, INT_MAX);
        dp[0] = 0;
        for (int i = 1; i <= n; ++i) {
            for (int j = 1; j * j <= i; ++j){
            dp[i] = min(dp[i], dp[i - j * j] + 1);
            }
        }
        return dp[n];
    }
};
```

Output:



```
Accepted   Runtime: 0 ms
• Case 1   • Case 2
Input
n =
12
Output
3
Expected
3
```

Case 1



```
Accepted   Runtime: 0 ms
• Case 1   • Case 2
Input
n =
13
Output
2
Expected
2
```

Case 2

Problem 5

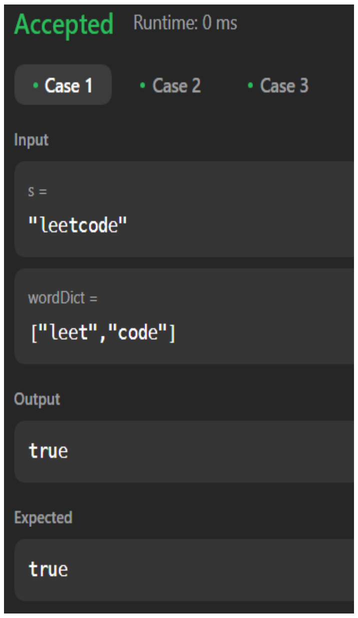
Aim:

Word Break

Code:

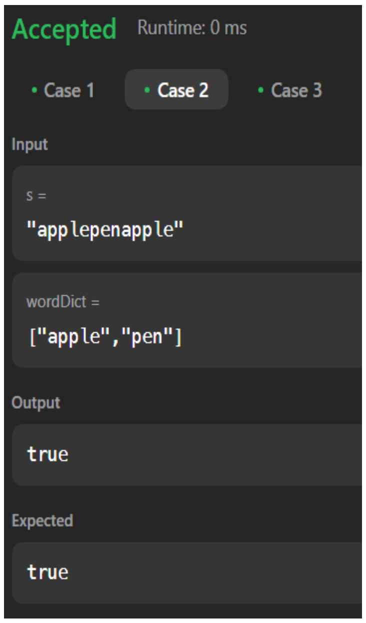
```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        vector<bool> dp(s.size() + 1, false);
        dp[0] = true;

        for (int i = 1; i <= s.size(); i++) {
            for (const string& w : wordDict) {
                int start = i - w.length();
                if (start >= 0 && dp[start] && s.substr(start, w.length()) == w) {
                    dp[i] = true;
                    break;
                }
            }
        }
        return dp[s.size()];
    }
};
```

Output:



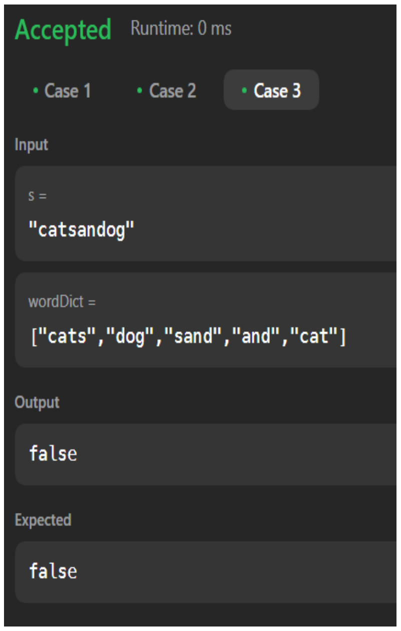
```
Accepted Runtime: 0 ms
• Case 1 • Case 2 • Case 3
Input
s =
"leetcode"
wordDict =
["leet","code"]
Output
true
Expected
true
```

Case 1



```
Accepted Runtime: 0 ms
• Case 1 • Case 2 • Case 3
Input
s =
"applepenapple"
wordDict =
["apple","pen"]
Output
true
Expected
true
```

Case 2



```
Accepted Runtime: 0 ms
• Case 1 • Case 2 • Case 3
Input
s =
"catsandog"
wordDict =
["cats","dog","sand","and","cat"]
Output
false
Expected
false
```

Case 3

Problem 6

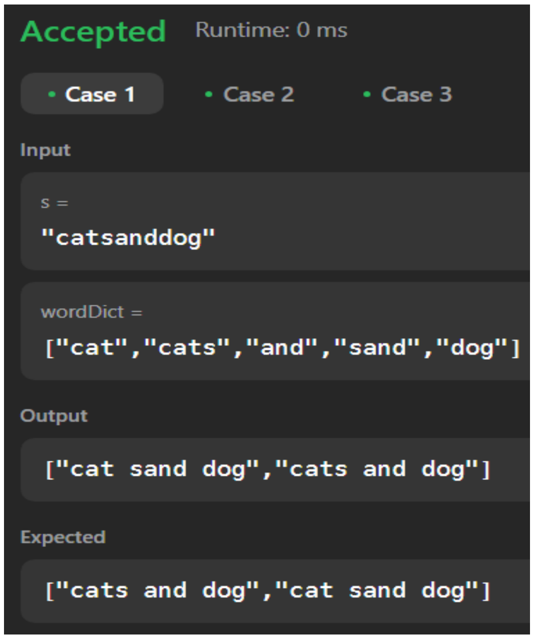
Aim:

Word Break 2

Code:

```
class Solution {
public:
    void solve(string s, vector<string>& res, unordered_set<string>& st, vector<string>&temp){
        if(s.length() == 0){
            string str = "";
            for(auto it:temp){
                str += it + " ";
            }
            str.pop_back();
            res.push_back(str);
            return;
        }
        for(int i=0;i<s.length(); i++){
            if(st.count(s.substr(0, i+1))){
                temp.push_back(s.substr(0, i+1));
                solve(s.substr(i+1), res, st, temp);
                temp.pop_back();
            }
        }
    }
    vector<string> wordBreak(string s, vector<string>& wordDict) {
        vector<string>res, temp;
        unordered_set<string>st(wordDict.begin(), wordDict.end());

        solve(s, res, st, temp);
        return res;
    }
};
```

Output:



Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =

```
"catsanddog"
```

wordDict =

```
["cat","cats","and","sand","dog"]
```

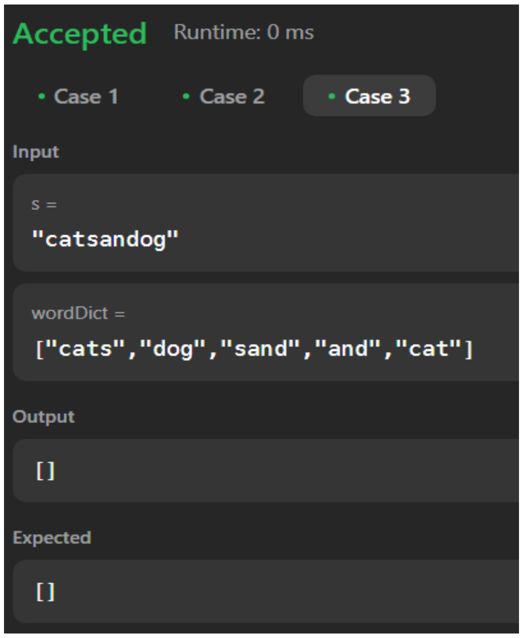
Output

```
["cat sand dog","cats and dog"]
```

Expected

```
["cats and dog","cat sand dog"]
```

Case 1



Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =

```
"catsandog"
```

wordDict =

```
["cats","dog","sand","and","cat"]
```

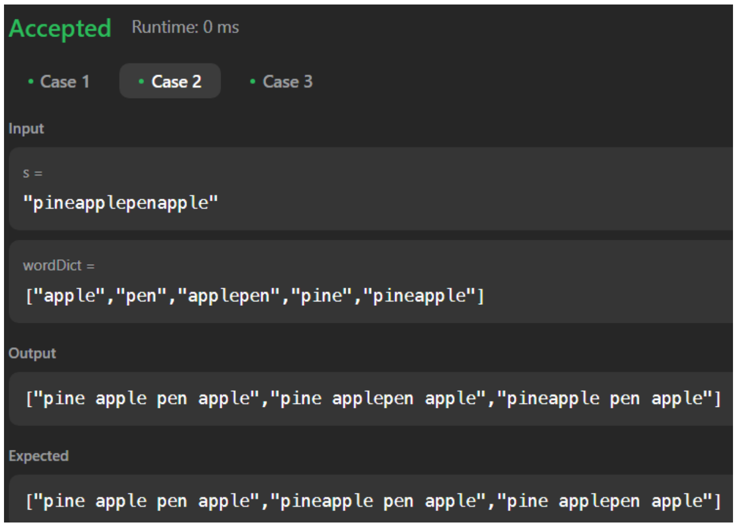
Output

```
[]
```

Expected

```
[]
```

Case 3



Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =

```
"pineapplepenapple"
```

wordDict =

```
["apple","pen","applepen","pine","pineapple"]
```

Output

```
["pine apple pen apple","pine applepen apple","pineapple pen apple"]
```

Expected

```
["pine apple pen apple","pineapple pen apple","pine applepen apple"]
```

Case 2

Problem 7

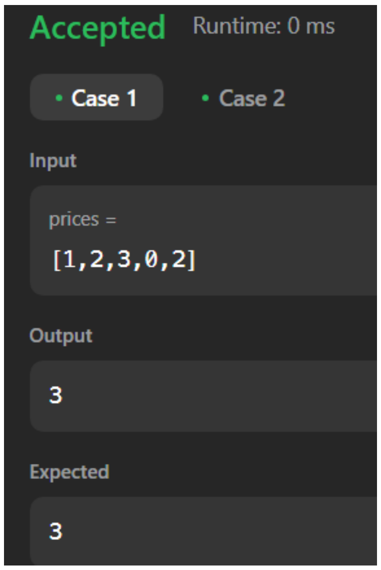
Aim:

Best time to buy and Sell a Stock with Cooldown

Code:

```
class Solution {
   public int maxProfit(int[] prices) {
      if (prices == null || prices.length <= 1) return 0;

      int b0 = -prices[0], b1 = b0;
      int s0 = 0, s1 = 0, s2 = 0;

      for (int i = 1; i < prices.length; i++) {
         b0 = Math.max(b1, s2 - prices[i]);
         s0 = Math.max(s1, b1 + prices[i]);
         b1 = b0;
         s2 = s1;
         s1 = s0;
      }
      return s0;
   }

   public static void main(String[] args) {
      Solution solution = new Solution();
      int[] prices = {1, 2, 3, 0, 2};  // Example input
      System.out.println("Max Profit: " + solution.maxProfit(prices));
   }
}
```

Output:



Input

```
prices =
[1,2,3,0,2]
```

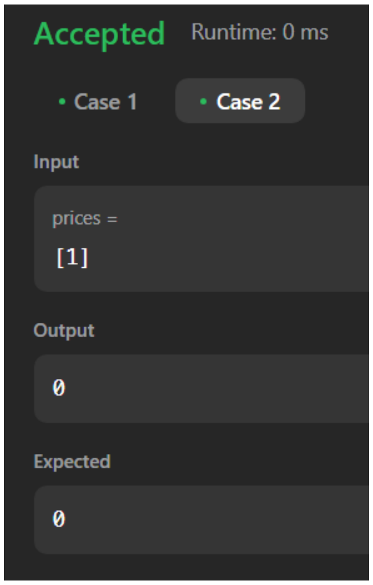
Output

```
3
```

Expected

```
3
```

Case 1



Input

```
prices =
[1]
```

Output

```
0
```

Expected

```
0
```

Case 2