Worksheet - 8

NAME: Manpreet Kaur **SEC-DWWC 43**

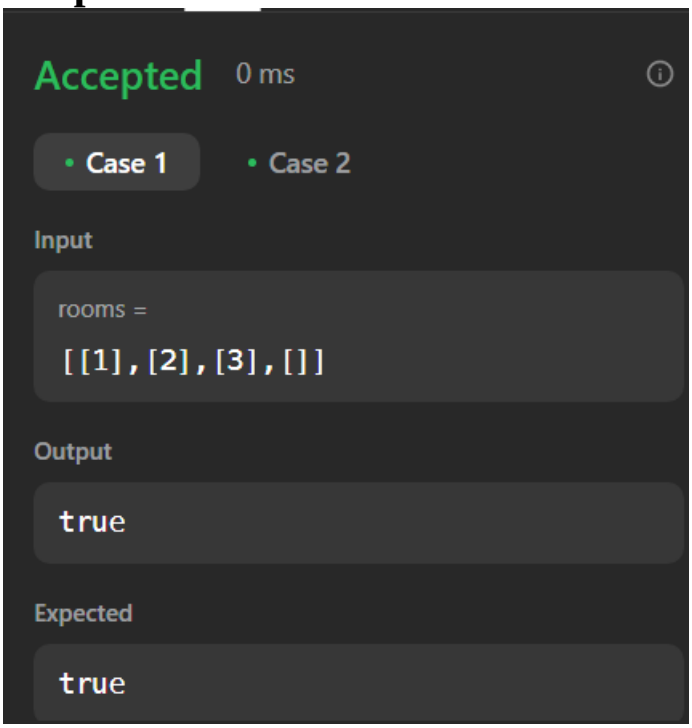
UID: 20BCS2212 **Date- 12/01/2023**

Que-1: Keys and Rooms

Code:

```
class Solution
{
public:
    bool canVisitAllRooms(vector<vector<int>>& rooms)
    {
        deque<int> Q({0});
        unordered_set<int> seen;
        while (!Q.empty())
        {
            int r = Q.front(); Q.pop_front();
            seen.insert(r);
            for (int k : rooms[r])
                if (!seen.count(k))
                    Q.push_back(k);
        }

        return seen.size() == rooms.size();
    }
};
```

Output:

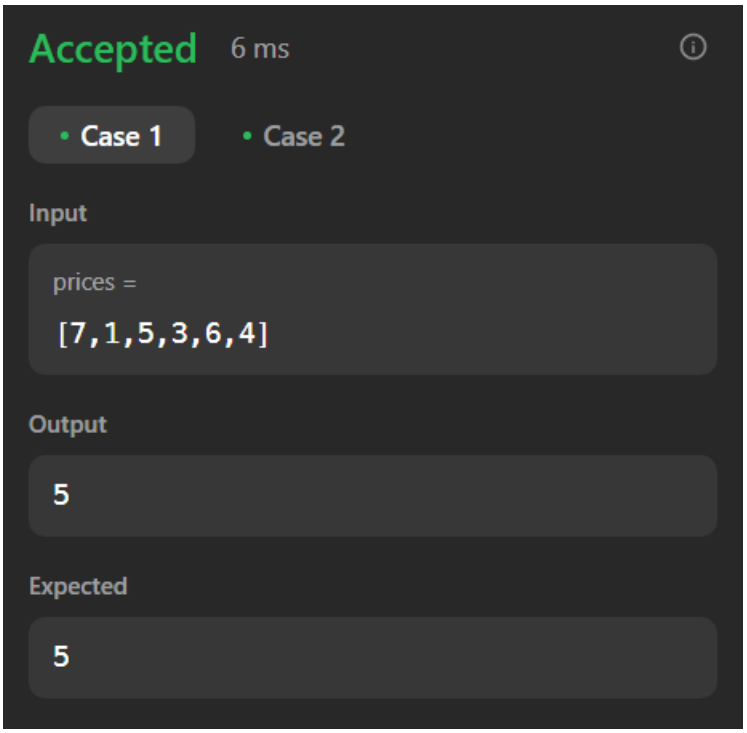


Que-2: Best Time to Buy and Sell Stock

Code:

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int max=0,min=prices[0];
        for (int i=0;i<prices.size();i++){
            int s=prices[i] - min;
            if(max<s){
                max=s;
            }
            if(prices[i]<min){
                min=prices[i];
            }
        }
        return max;
    }
};
```

Output:

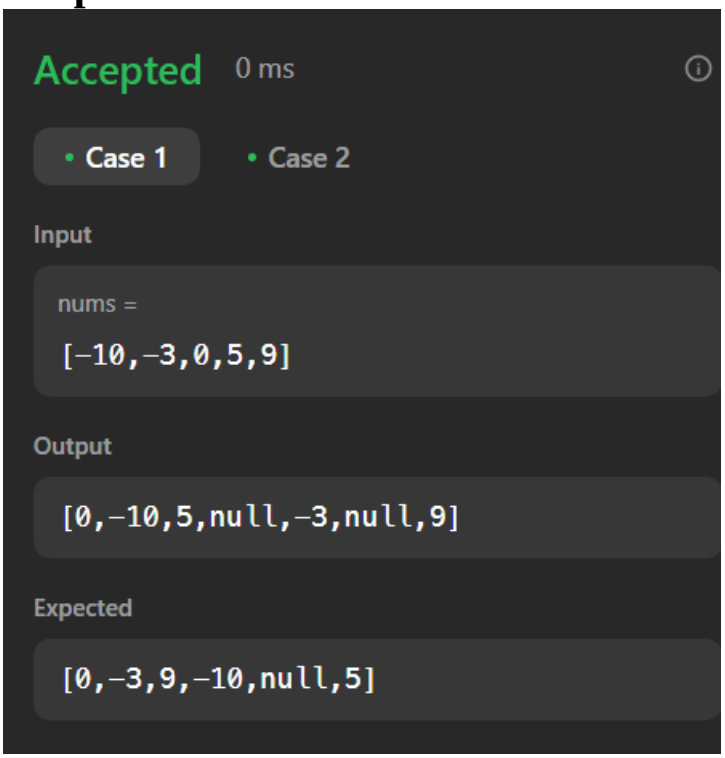


Que-3: Convert Sorted Array to Binary Search Tree

Code:

```
class Solution {
public:
    TreeNode* binarySearch(vector<int>& nums, int l, int r) {
        if(l > r) return NULL;
        int mid = l + (r-l)/2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = binarySearch(nums, l, mid-1);
        root->right = binarySearch(nums, mid+1, r);
        return root;
    }

    TreeNode* sortedArrayToBST(vector<int>& nums) {
        int n = nums.size();
        if(n == 0) return NULL;
        return binarySearch(nums, 0, n-1);

    }
};
```

Output:



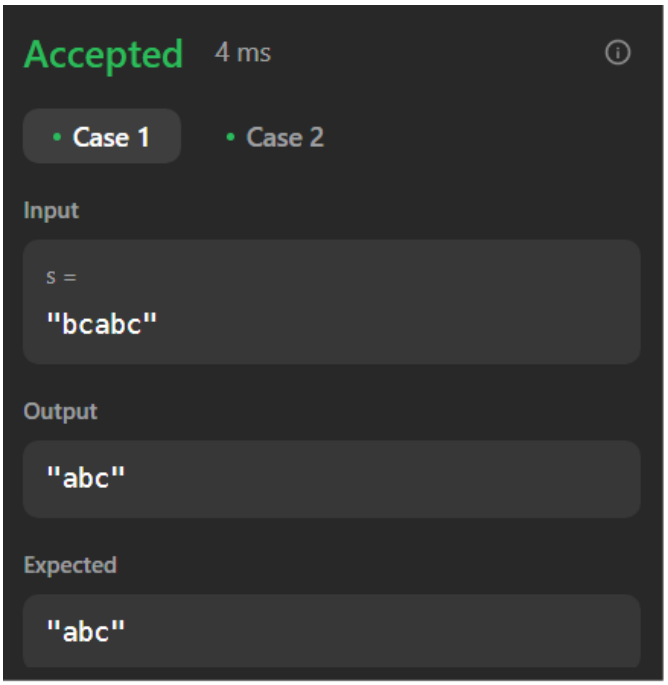
Que-4: [Remove Duplicate Letters]

Code:

```
class Solution {
public:
    string removeDuplicateLetters(string s) {
        int len = s.size();
        string res = "";
        unordered_map<char, int> M;
        unordered_map<char, bool> V;
        stack<int> S;

        for (auto c : s) {
            if (M.find(c) == M.end()) M[c] = 1;
            else M[c]++;
        }
        for (unordered_map<char, int>::iterator iter=M.begin();
iter!=M.end(); iter++) V[iter->first] = false;

        cout<<M.size()<<V.size()<<endl;
        for (int i=0; i<len; i++) {
```

```
            M[s[i]]--;
            if (V[s[i]] == true) continue;

            while (!S.empty() and s[i] < s[S.top()] and M[s[S.top()]]
> 0) {
                V[s[S.top()]] = false;
                S.pop();
            }
            S.push(i);
            V[s[i]] = true;
        }
        while (!S.empty()) {
            res = s[S.top()] + res;
            S.pop();
        }
        return res;
    }
};
```

Output:

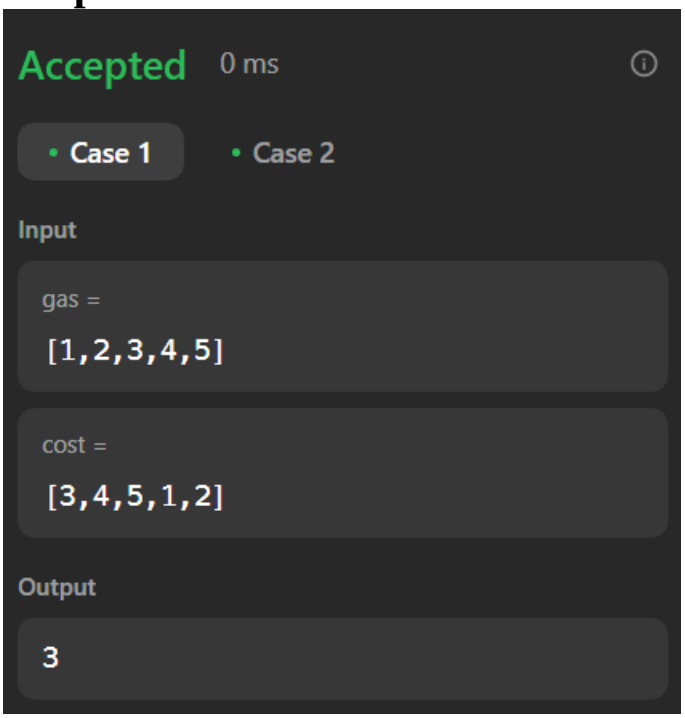


Que-5: Gas Station

Code:

```
class Solution {
public:
    int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {
        int n=gas.size();
        int total_gas=0,total_cost=0;
        int curr_gas=0, starting_point=0;
        for(int i=0;i<n;i++)
        {
            total_gas+=gas[i];
            total_cost+=cost[i];

            curr_gas+=gas[i]-cost[i];
            if(curr_gas<0)
            {
                starting_point=i+1;
                curr_gas=0;
            }
        }
        return (total_gas<total_cost)?-1:starting_point;
    }
};
```

Output:



Que-6: [Divide a String Into Groups of Size k]

Code:

```
class Solution {
public:
    vector<string> divideString(string s, int k, char fill) {
        vector<string> ans;
        string temp="";
        for(auto i:s){
            temp+=i;
            if(temp.size()==k){
                ans.push_back(temp);
                temp="";
            }
        }
        if(temp.size()!=k and temp!=""){
            while(temp.size()<k){
                temp+=fill;
            }
            ans.push_back(temp);
        }
        return ans;
    }
};
```

Output:

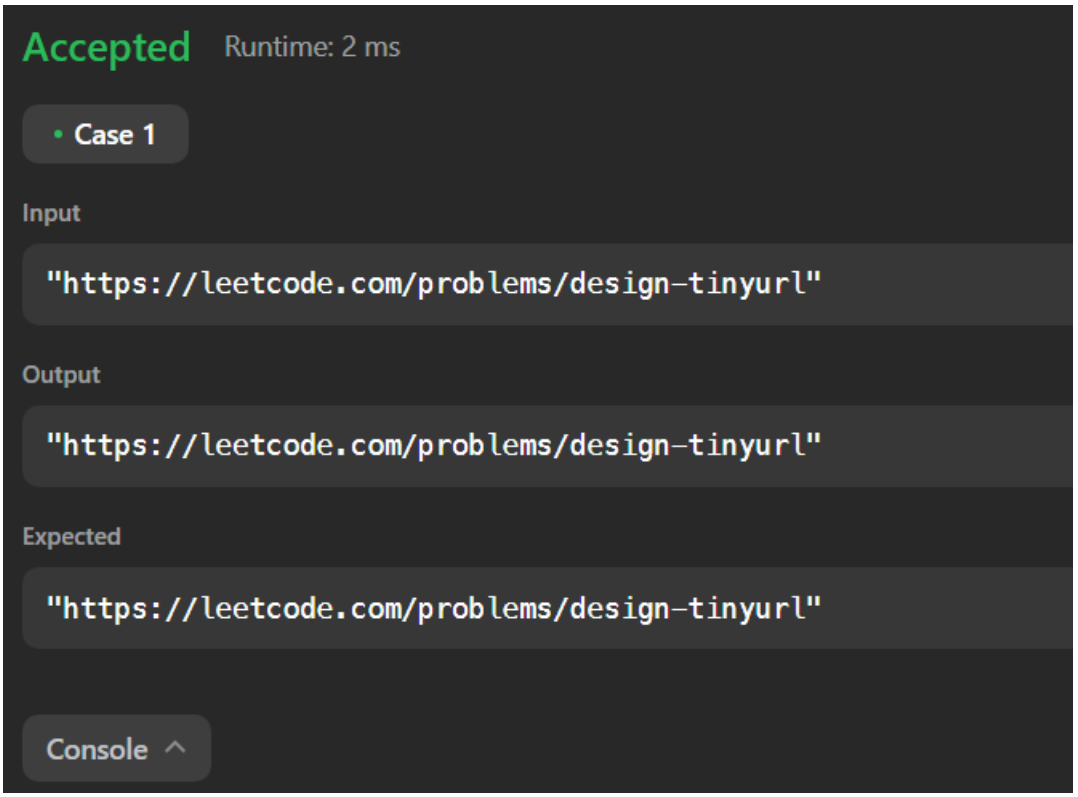
Que-7: [Encode and Decode TinyURL]

Code:

```
class Solution {
public:

    // Encodes a URL to a shortened URL.
    unordered_map<string,string> map;
    int n=0;
    string encode(string longUrl) {
        n++;
        string res="http://tinyurl.com/";
        res+=to_string(n);
        map[res]=longUrl;
        return res;
```

```
    }

    // Decodes a shortened URL to its original URL.
    string decode(string shortUrl) {
        return map[shortUrl];
    }
};
```

Output:



Que-8: [Check if Number Has Equal Digit Count and Digit Value]

Code:

```
class Solution {
public:
    bool digitCount(string nums) {
        unordered_map<int,int> map;
        for(auto i: nums){
            int t=int(i) - 48;
            map[t]++;
```

```
        }
        for(int i=0;i<nums.size();i++){
            int t=int(nums[i]) - 48;
            if(t!= map[i]){
                return false;
            }
        }
        return true;
    }
};
```

Output: