NAME: MOAZZAM FAROOQUI

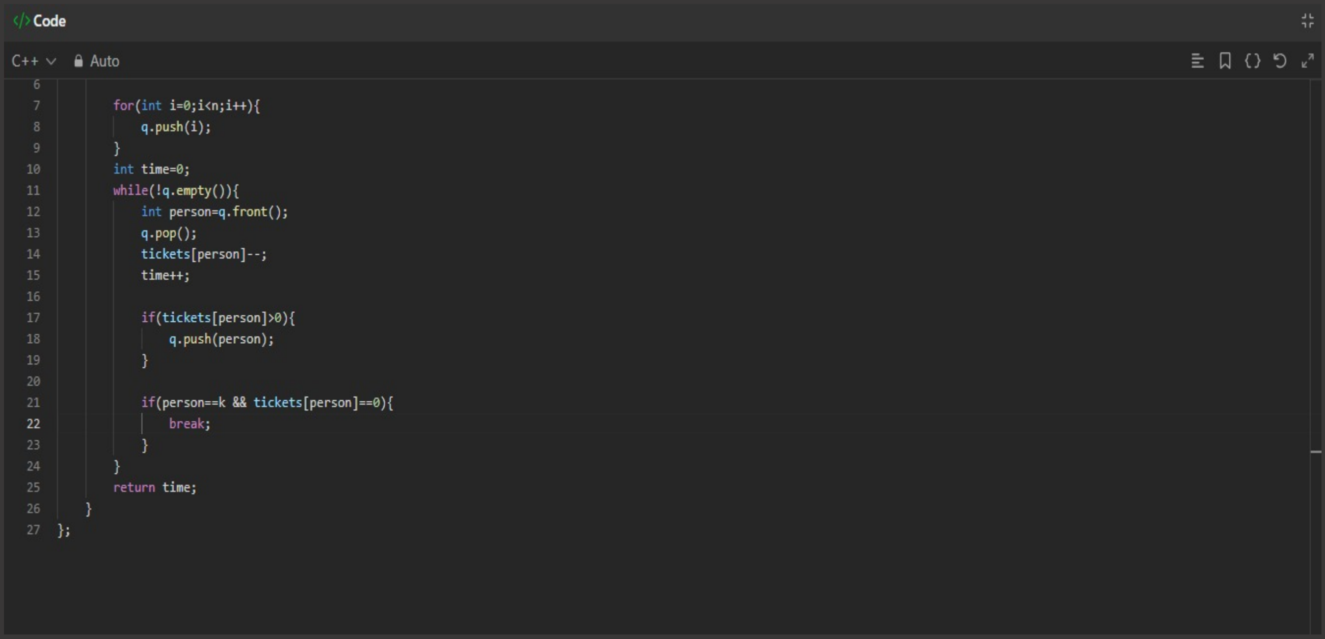
ROLL NO: CT-24068

COURSE CODE: CT-159

ASSIGNMENT: DSA PRACTICE SESSION TASK#4

Q1. Time Needed to Buy Tickets

SOURCE CODE:



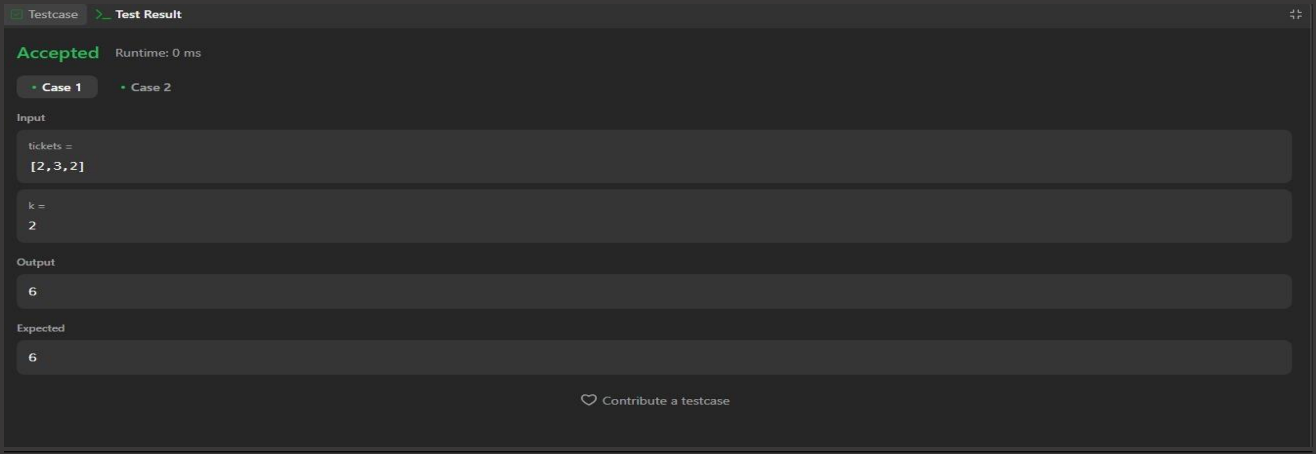
```cpp

        for(int i=0;i<n;i++){
            q.push(i);
        }
        int time=0;
        while(!q.empty()){
            int person=q.front();
            q.pop();
            tickets[person]--;
            time++;

            if(tickets[person]>0){
                q.push(person);
            }

            if(person==k && tickets[person]==0){
                break;
            }
        }
        return time;
    }
};
```

ACCEPTANCE STATUS:



Accepted Runtime: 0 ms

Input

tickets =
[2,3,2]

k =
2

Output

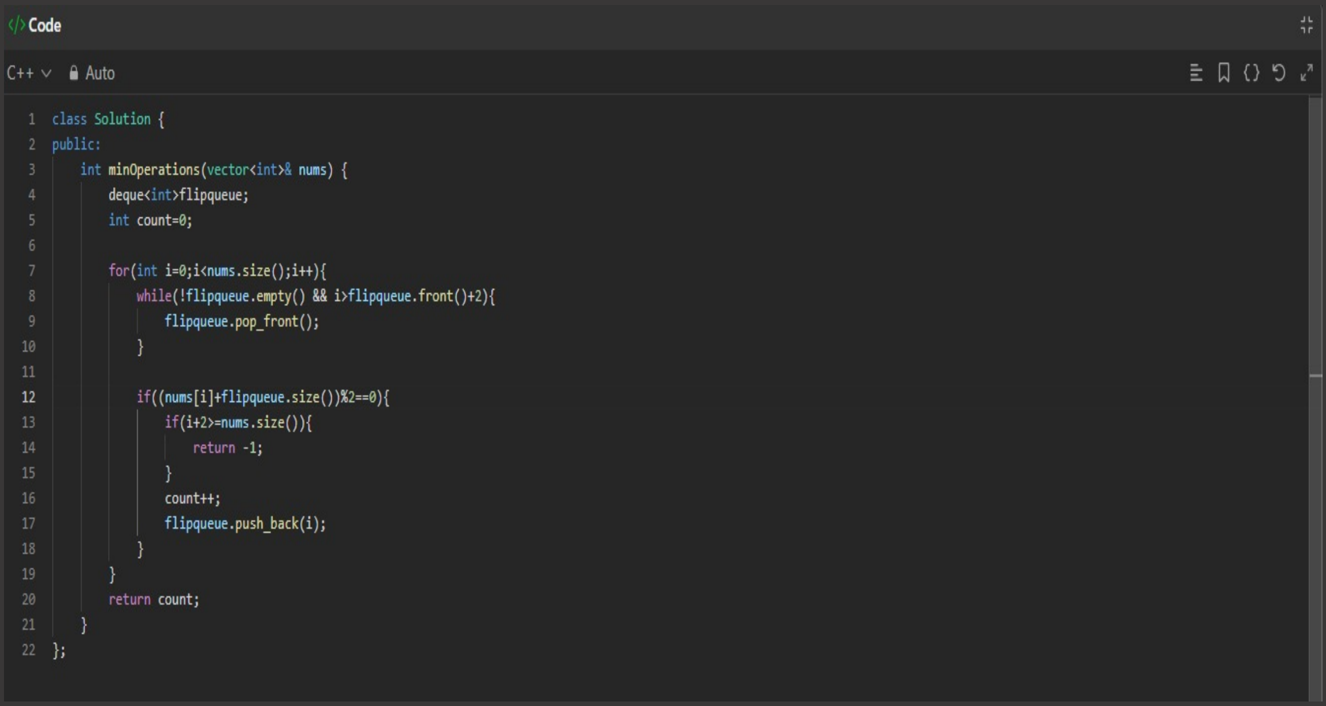
6

Expected

6

Q2. Minimum Operations to Make Binary Array Elements Equal to One

SOURCE CODE:

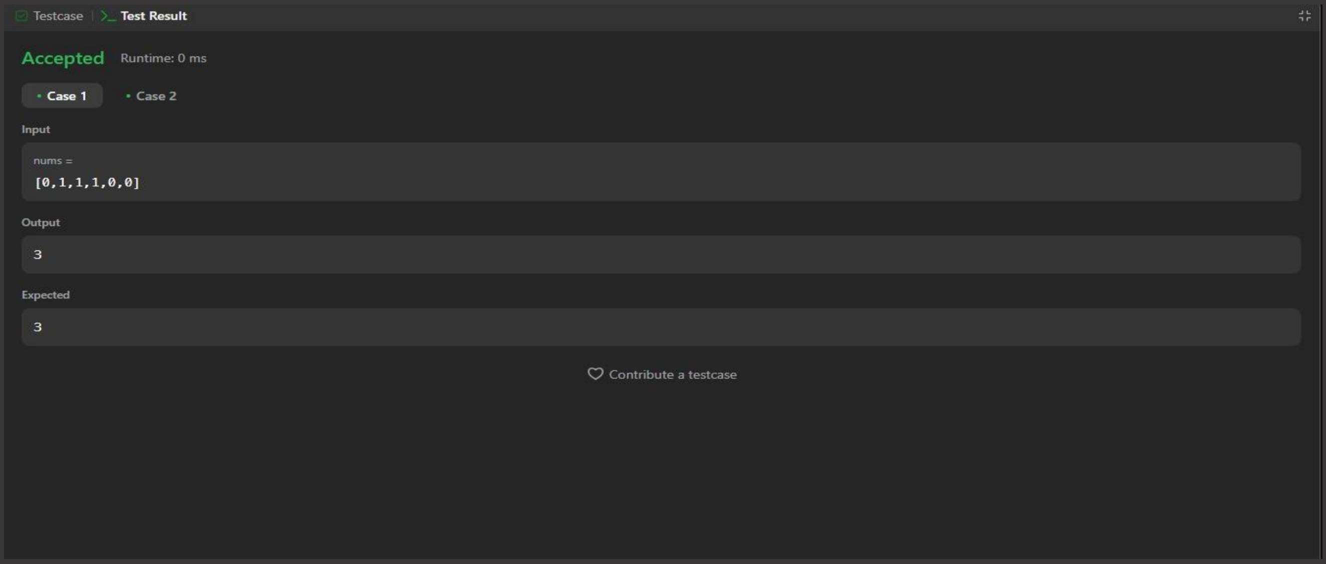


```cpp
class Solution {
public:
    int minOperations(vector<int>& nums) {
        deque<int>flipqueue;
        int count=0;

        for(int i=0;i<nums.size();i++){
            while(!flipqueue.empty() && i>flipqueue.front()+2){
                flipqueue.pop_front();
            }

            if((nums[i]+flipqueue.size())%2==0){
                if(i+2>=nums.size()){
                    return -1;
                }
                count++;
                flipqueue.push_back(i);
            }
        }
        return count;
    }
};
```

ACCEPTANCE STATUS:



Accepted Runtime: 0 ms

Input

```
nums =
[0,1,1,1,0,0]
```

Output

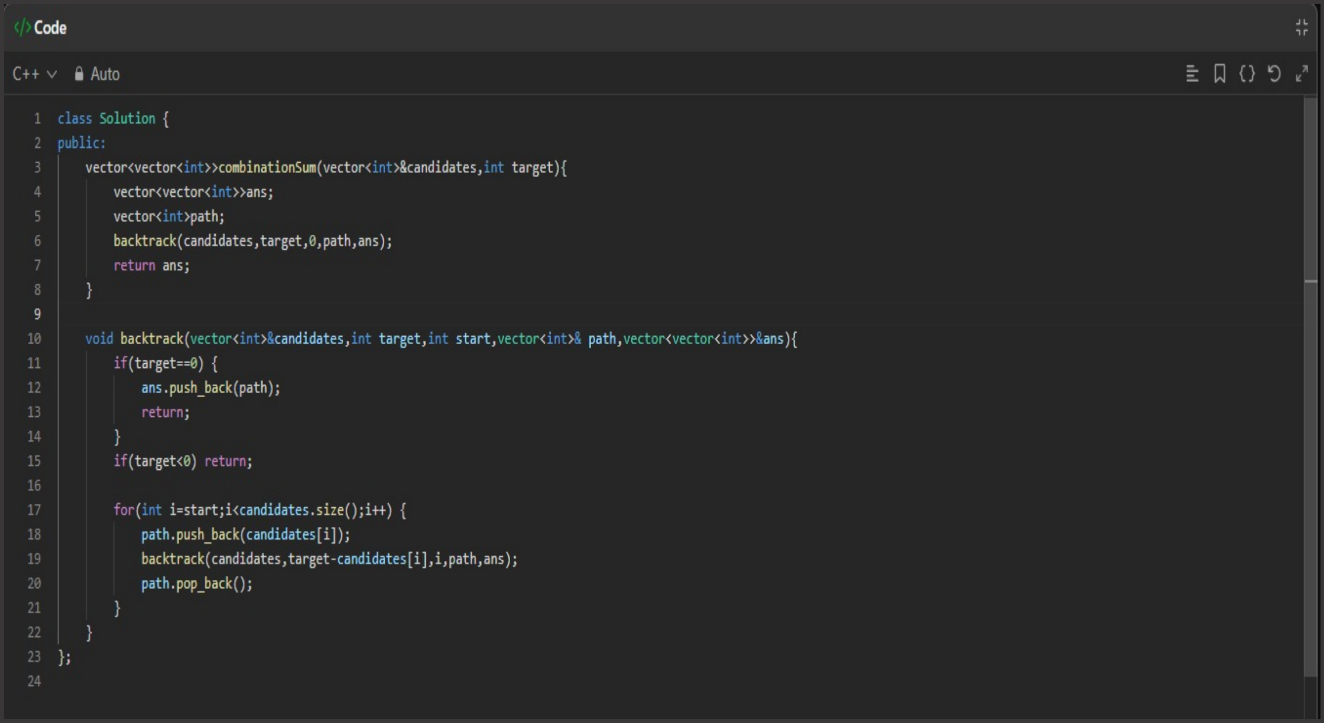
```
3
```

Expected

```
3
```

Q3. Combination Sum

SOURCE CODE:

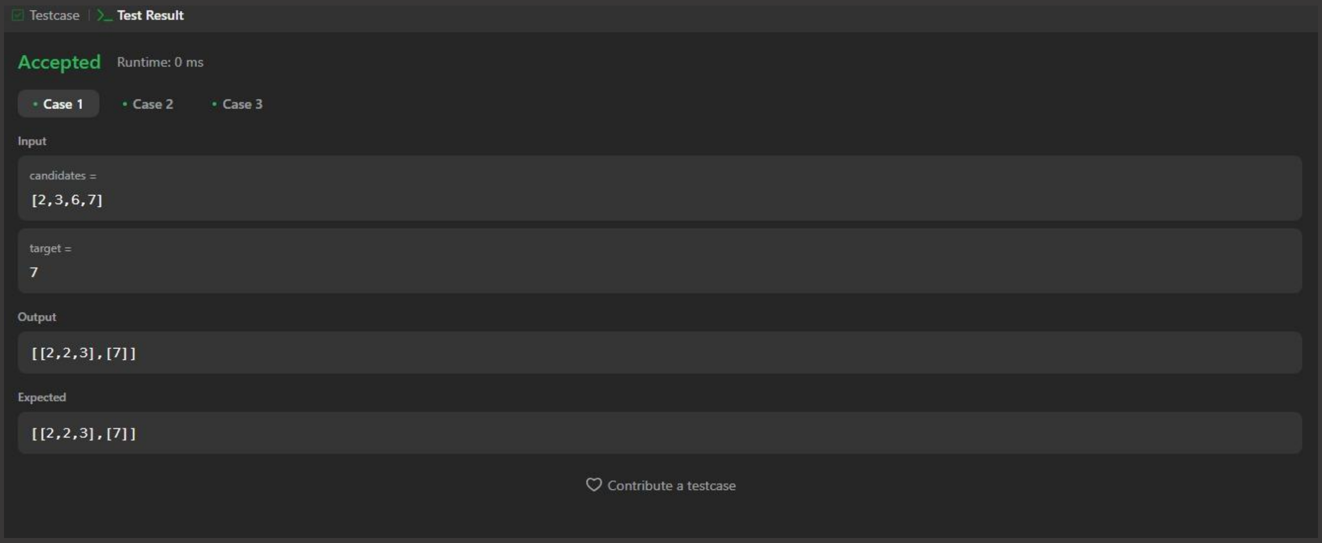


```cpp
class Solution {
public:
    vector<vector<int>>combinationSum(vector<int>&candidates,int target){
        vector<vector<int>>ans;
        vector<int>path;
        backtrack(candidates,target,0,path,ans);
        return ans;
    }

    void backtrack(vector<int>&candidates,int target,int start,vector<int>& path,vector<vector<int>>&ans){
        if(target==0) {
            ans.push_back(path);
            return;
        }
        if(target<0) return;

        for(int i=start;i<candidates.size();i++) {
            path.push_back(candidates[i]);
            backtrack(candidates,target-candidates[i],i,path,ans);
            path.pop_back();
        }
    }
};
```

ACCEPTANCE STATUS:



Accepted Runtime: 0 ms

Input

candidates =
```
[2,3,6,7]
```

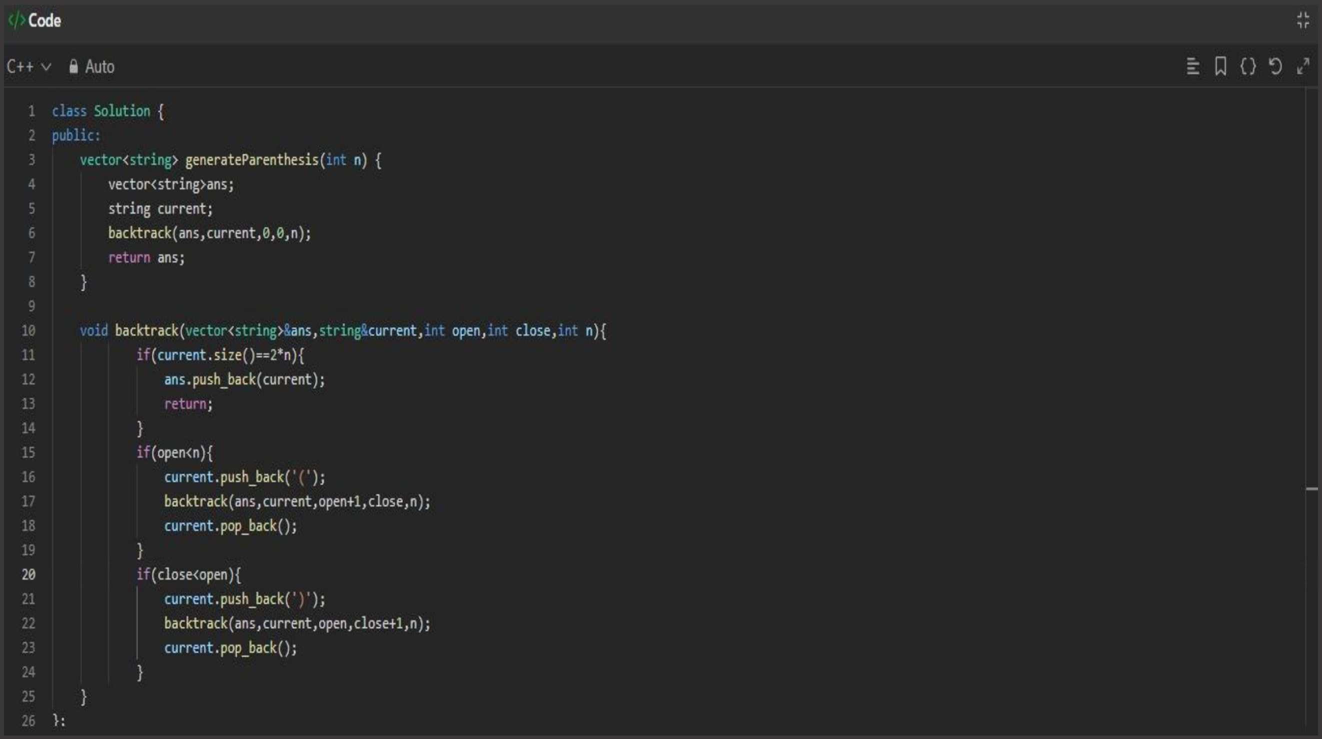
target =
```
7
```

Output
```
[[2,2,3],[7]]
```

Expected
```
[[2,2,3],[7]]
```

Q4. Generate Parenthesis

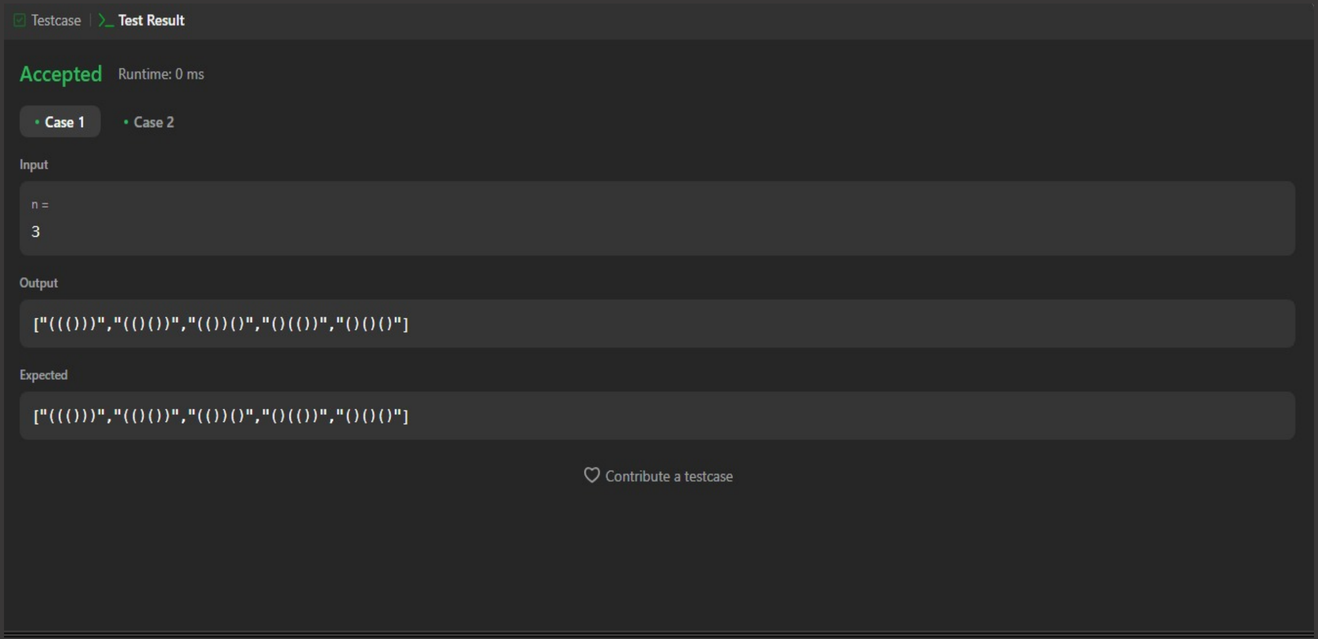
SOURCE CODE:



```cpp
class Solution {
public:
    vector<string> generateParenthesis(int n) {
        vector<string>ans;
        string current;
        backtrack(ans,current,0,0,n);
        return ans;
    }

    void backtrack(vector<string>&ans,string&current,int open,int close,int n){
            if(current.size()==2*n){
                ans.push_back(current);
                return;
            }
            if(open<n){
                current.push_back('(');
                backtrack(ans,current,open+1,close,n);
                current.pop_back();
            }
            if(close<open){
                current.push_back(')');
                backtrack(ans,current,open,close+1,n);
                current.pop_back();
            }
    }
};
```

ACCEPTANCE STATUS:



Accepted Runtime: 0 ms

Input

```
n =
3
```

Output

```
["((()))","(()())","(())()","()(())","()()()"]
```

Expected

```
["((()))","(()())","(())()","()(())","()()()"]
```