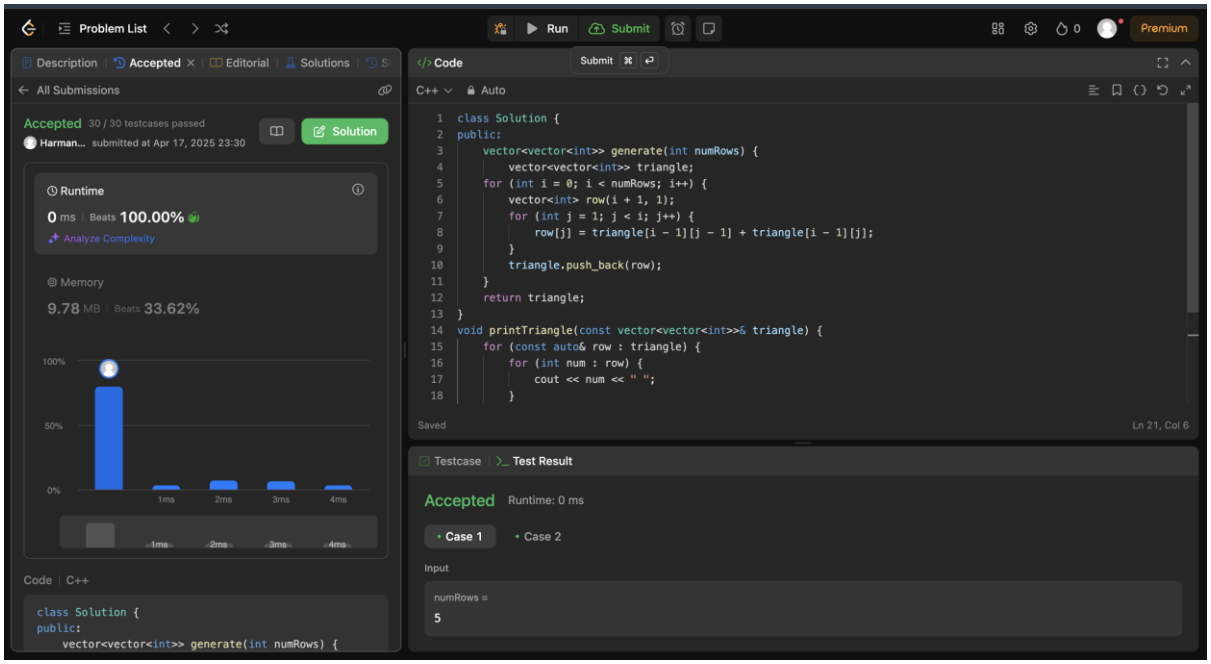
AP-10

Name: Harmandeep Singh **Section:614-B**

UID:22BCS14975

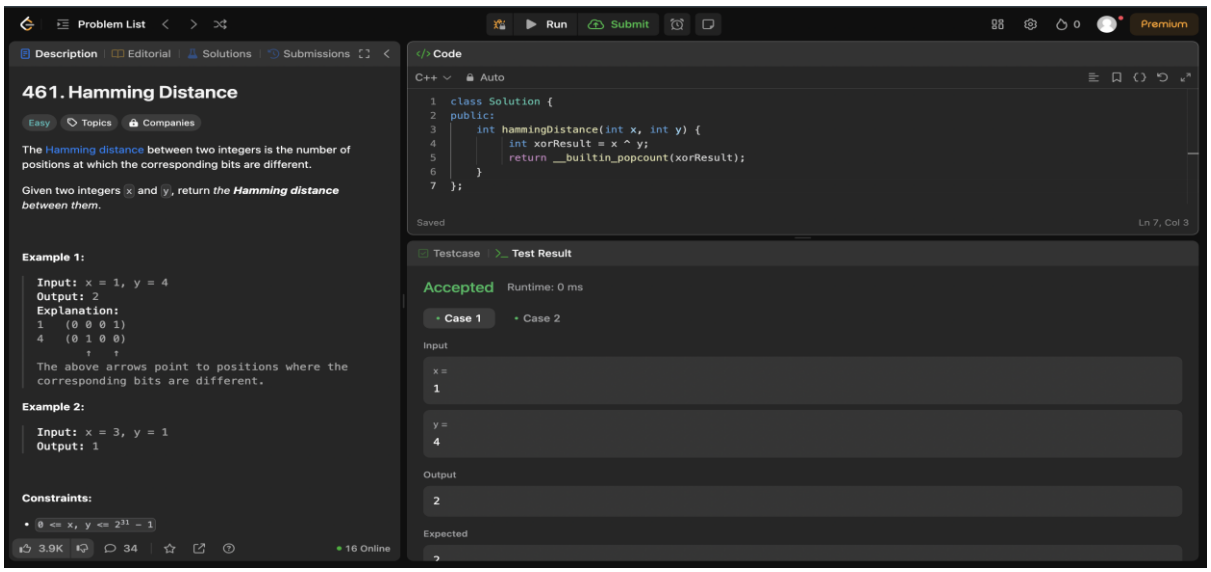
1.Pascal's Triangle

```
class Solution {
public:
vector<vector<int>> generate(int numRows) {
vector<vector<int>> triangle;
for (int i = 0; i < numRows; i++) {
vector<int> row(i + 1, 1);
for (int j = 1; j < i; j++) {
row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
}
triangle.push_back(row);
}
return triangle;
}
void printTriangle(const vector<vector<int>>& triangle) {
for (const auto& row : triangle) {
for (int num : row) {
cout << num << " ";
}
cout << endl;
}
}
};
```



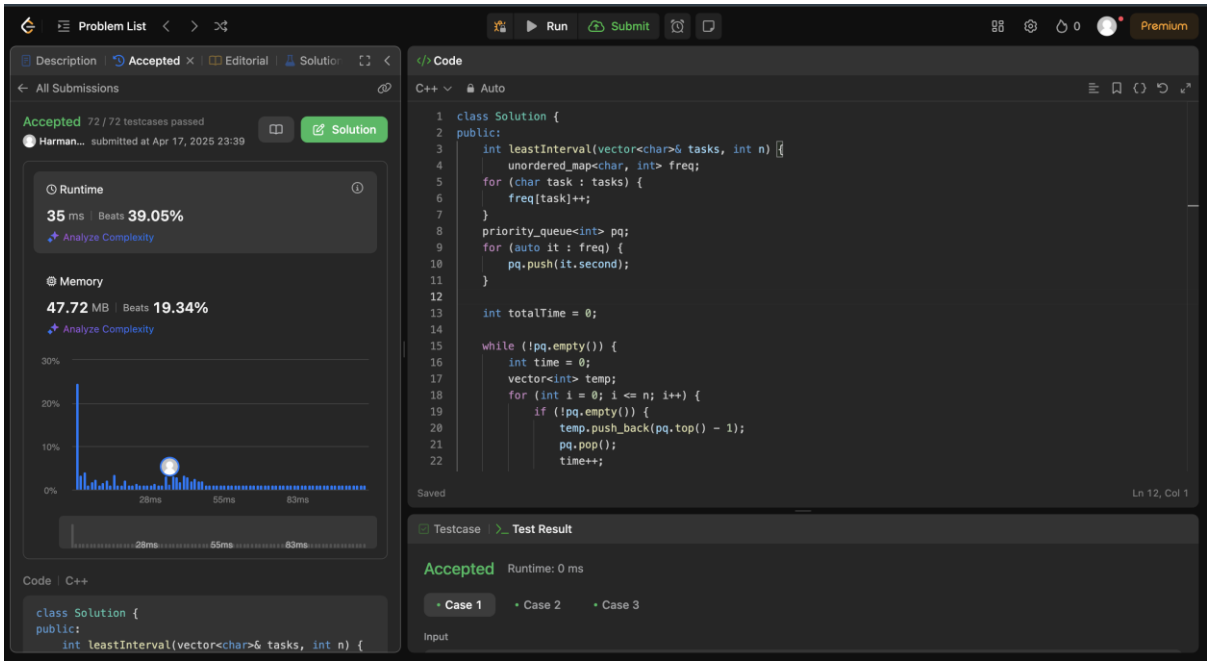
2.Hamming Distance

```
class Solution {
public:
int hammingDistance(int x, int y) {
int xorResult = x ^ y;
return __builtin_popcount(xorResult);
}
};
```



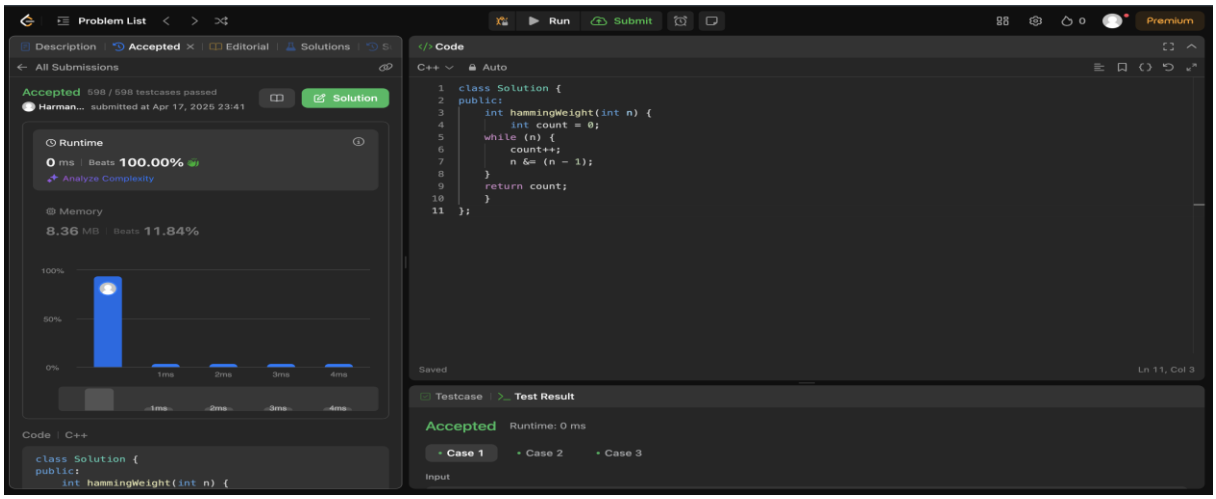
3.Task Schedular

```
class Solution {
public:
int leastInterval(vector<char>& tasks, int n) {
unordered_map<char, int> freq;
for (char task : tasks) {
freq[task]++;
}
priority_queue<int> pq;
for (auto it : freq) {
pq.push(it.second);
}

int totalTime = 0;

while (!pq.empty()) {
int time = 0;
vector<int> temp;
for (int i = 0; i <= n; i++) {
if (!pq.empty()) {
temp.push_back(pq.top() - 1);
pq.pop();
time++;
}
}

for (int count : temp) {
if (count > 0) pq.push(count);
}

totalTime += pq.empty() ? time : (n + 1);
}

return totalTime;
}
};
```



4.Number of 1‘s bits

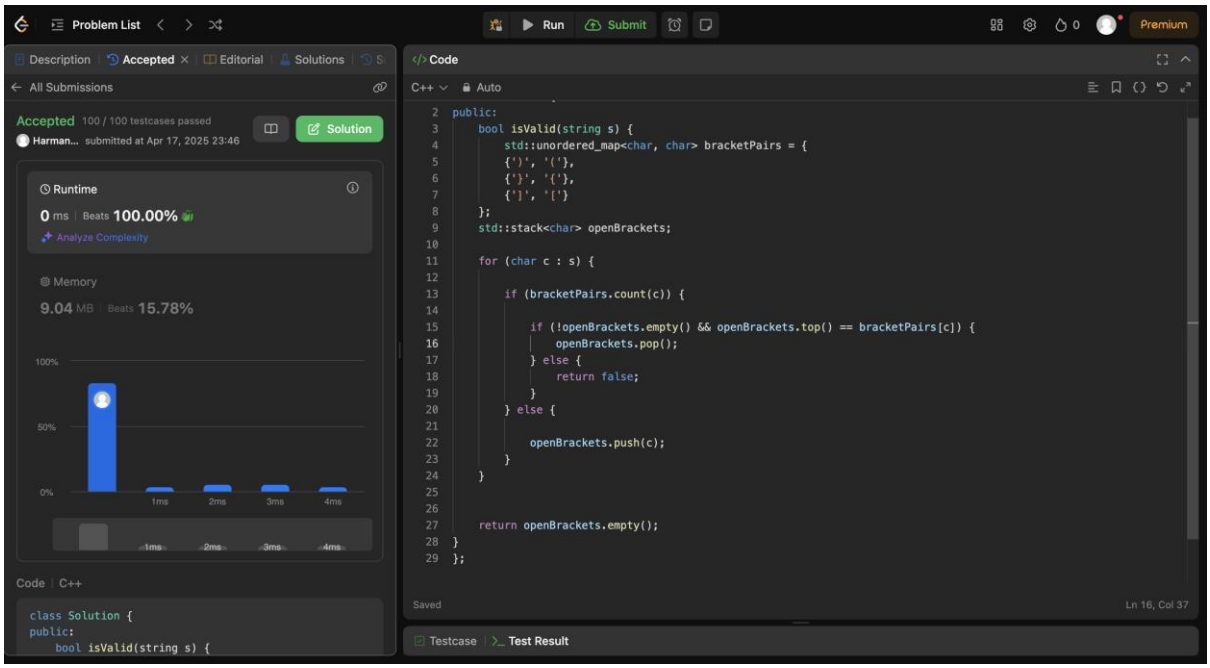
```
class Solution {
public:
int hammingWeight(int n) {
int count = 0;
while (n) {
count++;
n &= (n - 1);
}
return count;
}
};
```



5.Valid Parenthesis

```
public:
```

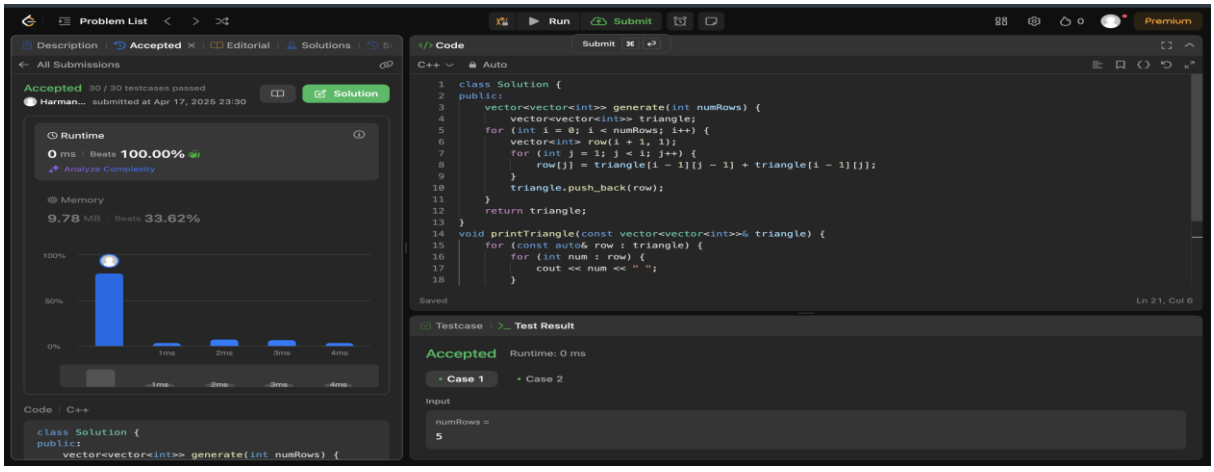
```
bool isValid(string s) {
std::unordered_map<char, char> bracketPairs = {
{')', '('},
{'}', '{'},
{']', '['}
};
std::stack<char> openBrackets;

for (char c : s) {
if (bracketPairs.count(c)) {
if (!openBrackets.empty() && openBrackets.top() == bracketPairs[c]) {
openBrackets.pop();
} else {
return false;
}
} else {
openBrackets.push(c);
}
}

return openBrackets.empty();
}
};
```



6.Divide Two Integer

```
class Solution {
```

```
public:
int divide(int dividend, int divisor) {
if (dividend == INT_MIN && divisor == -1)
return INT_MAX;

long long a = abs((long long)dividend);
long long b = abs((long long)divisor);
int result = 0;

while (a >= b) {
long long temp = b, multiple = 1;

while (a >= (temp << 1)) {
temp <<= 1;
multiple <<= 1;
}

a -= temp;
result += multiple;
}

if ((dividend > 0) ^ (divisor > 0))
result = -result;

return result;
}
};
```



7.Trapping Rain Water

```
class Solution {
public:
int trap(vector<int>& height) {
int left = 0, right = height.size() - 1;
```

```
int leftMax = 0, rightMax = 0;
int waterTrapped = 0;


while (left < right) {
if (height[left] < height[right]) {
if (height[left] >= leftMax)
leftMax = height[left];
else
waterTrapped += leftMax - height[left];
left++;
} else {
if (height[right] >= rightMax)
rightMax = height[right];
else
waterTrapped += rightMax - height[right];
right--;
}
}

return waterTrapped;
}
};
```